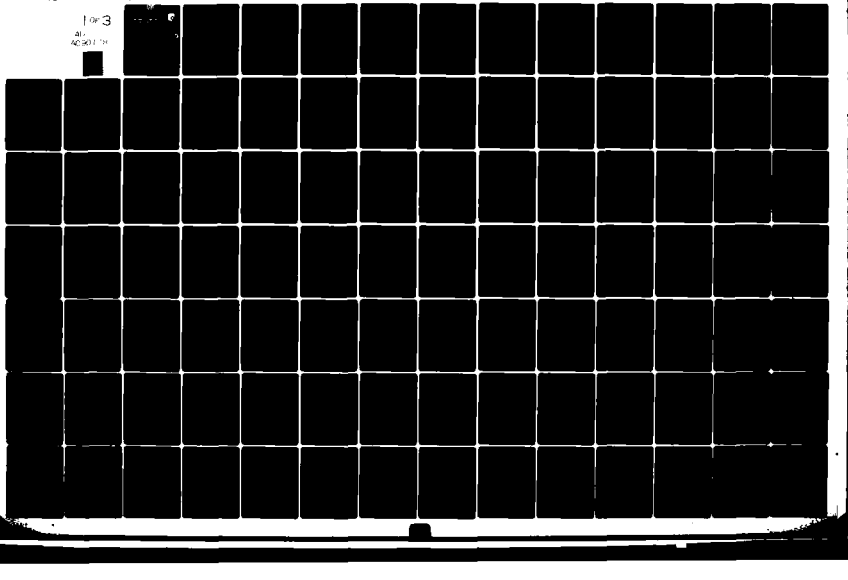
AD-A090 178 HONEYWELL SYSTEMS AND RESEARCH CENTER MINNEAPOLIS MN F/G 14/2
UNIT UNDER TEST SIMULATOR FEASIBILITY STUDY.(U)
JUN 80 S J NUSPL, D T LEE, J P HUDSON F33615-79-C-1811
UNCLASSIFIED 80SRC46 AFWAL-TR-80-1091 NL



LEVEL II

(2)

F

AFWAL-TR-80-1091

UNIT UNDER TEST SIMULATOR FEASIBILITY STUDY

AD A090178

Honeywell

SYSTEMS & RESEARCH CENTER

2600 RIDGWAY PARKWAY
MINNEAPOLIS, MINNESOTA 55413

DTIC ELECTE OCT 9 1980 S D C

FINAL TECHNICAL REPORT

June 1980

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

80 [illegible] 5

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Nelson N Estes

NELSON N. ESTES
Project Engineer

Raymond D. Bellem

RAYMOND D. BELLEM, Lt Col, USAF
Chief, Support Systems Branch
Avionics Laboratory

FOR THE COMMANDER

Raymond E Siferd

RAYMOND E. SIFERD, Col, USAF
Chief, System Avionics Division
Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAF, *W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/24 September 1980 — 80

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER: AFWAL-TR-80-1091	2 GOVT ACCESSION NO.: AD-A090178
	3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle): UNIT UNDER TEST SIMULATOR FEASIBILITY STUDY. FINAL TECHNICAL REPORT	5 TYPE OF REPORT & PERIOD COVERED: Final Technical Report
	6 PERFORMING ORG. REPORT NUMBER: 80SRC46
7 AUTHOR(s): Stephen J. Nuspl, John P. Hudson, David T. Lee, Todd C. Steeves	8 CONTRACT OR GRANT NUMBER(s): WPAFB Contract No. F33615-79-C-1811 NEW
9 PERFORMING ORGANIZATION NAME AND ADDRESS: Honeywell Inc Systems and Research Center, 2600 Ridgway Parkway, Minneapolis, Minnesota 55413	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 2003-02-50
11 CONTROLLING OFFICE NAME AND ADDRESS: United States Air Force, Air Force Avionics Laboratory, Wright-Patterson AFB, OH 45433	12 REPORT DATE: June 1980
	13 NUMBER OF PAGES: 276 pages
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15 SECURITY CLASS. (of this report): Unclassified
	15a DECLASSIFICATION DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

UUT
Unit Under Test
Simulator
Automatic Test Equipment
MATE
Verification and Validation

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report addresses the use of a Unit Under Test (UUT) simulator as a potential solution for supporting the verification and validation (V&V) of test programs for automatic test equipment (ATE). The objectives are to increase the effectiveness of the verification and validation and to reduce development costs. The study approach involved a top-down look at the nature of the problem and a bottom-up look at the techniques and

(continued)

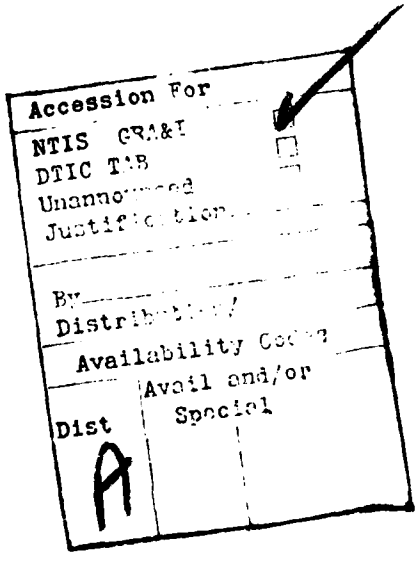
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (continued)

technologies that could be applied to realize a UUT simulator. The recommended solution is based on multiple levels of simulation and uses either an instrument bus interface to the test equipment or a software-only-based interface to the test-program interpreter. Further issues of cost-effectiveness must be addressed for the Modular Automatic Test Equipment (MATE) context.



CONTENTS

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (concluded)

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF TABLES (concluded)

LIST OF ACRONYMS

AATPG	Analog Automatic Test Program Generation
AIS	Avionics Intermediate Shop
ATE	Automatic Test Equipment
ATG	Automatic Test Generation
ATLAS	Abbreviated Test Language for Avionic Systems
ATPG	Automatic Test Program Generation
ATS	Automatic Testing System
CAD	Computer Aided Design
CAP	Circuit Analysis Program
COMPSIG	Complementary Signals
CUT	Circuit Under Test
DATPG	Digital Automatic Test Program Generation
DoD	Department of Defense
FI	Fault Isolation
HATS	Hybrid Automatic Test System
HDL	Hardware Discription Language
HOL	High Order Language
IA	Interface Adapter
ID	Interface Device (Interface Adapter)
ILASS	Intermediate Level Avionic Support System
ITA	Interface Test Adapter
LRU	Line Replaceable Unit
MATE	Modular Automatic Test Equipment
OIT	Operator Interface Terminal

LIST OF ACRONYMS (concluded)

OPAL	Operational Performance Analysis Language
PIU	Programmable Interface Unit
RATEL	Raytheon Automatic Test Equipment Language
RTL	Register Transfer Language
SIP	Standard Interface Package
SPICE	Simulation Program with Integrated Circuit Emphasis
SRA	Shop Replaceable Assembly
SRU	Shop Replaceable Unit
TEEM	Test Equipment Effectiveness Model
TPS	Test Package Set (also Test Program Set)
TRD	Test Requirement Document
UUT	Unit Under Test
V&V	Verification and Validation
VAST	Versatile Avionics Shop Test
VCOMP	VAST Compatibility Program
WRA	Weapon Replaceable Assembly

SECTION 1

INTRODUCTION AND SUMMARY

The development and maintenance of test programs have become one of the main cost factors in Air Force weapons systems. One of the major components of these development costs is Verification and Validation (V&V). This report addresses the use of a Unit Under Test (UUT) simulator as a potential solution in reducing V&V costs and in increasing the effectiveness of the V&V.

The typical approach during test program V&V is to execute the manually-generated test programs on the Automatic Test Equipment (ATE), using a real UUT to verify that the programs operate as intended. The following are some typical problems which are encountered:

- UUT unavailability during the V&V phase
- Delay of test program verification because the spare UUTs being used are pulled back for higher priority needs
- Delays because of UUT damage due to errors in the test programs
- Uncertainty of test-program coverage because fault paths cannot be forced in the UUT being used for V&V.

It was determined by the Air Force that one potential solution to the above problems is to use a UUT simulator rather than a real UUT during

as much of the V&V activity as possible. To assess the merit of this approach, the Air Force initiated the UUT Simulator Feasibility Investigation Study Program. This report covers the findings in the performance of this program.

The study approach basically involved a top-down look at the problem to be solved and a bottom-up look at the techniques and technologies that could be applied to realize a UUT simulator. During the first part of the program the following top-down activities were performed:

- Requirements were collected from the literature, from a validator survey, from analyzing Test Requirement Documents (TRDs), and from analyzing representative ATLAS test programs.
- A baseline set of requirements was established.
- A set of enhancement features was defined and prioritized.
- UUT simulator usage scenarios were postulated.
- Potential Interface Adapter (IA) verification usage was defined.

The specifics of the above are covered in Sections 2 and 3 of this report.

As a part of the bottom-up activities, the following were performed:

- The literature was searched for potentially applicable simulation techniques.
- Extensions of the techniques and specific sets of UUT simulator solutions were postulated.
- Trade-off information was derived for the classes of UUT simulators.

- Based on this information, recommended solutions were derived.
- Additional uses for a UUT simulator aside from supporting V&V were considered.

The results of the above are covered in Sections 4 and 5 of this report.

The following set of conclusions summarizes the outcome of the UUT simulator feasibility study program:

- The feasibility of simulators for classes of UUTs in the digital, analog, and hybrid area is clear; their cost-effectiveness in V&V must still be validated.
- To be most effective the UUT simulator development must start early in the development cycle of a Line Replaceable Unit (LRU), and the simulator must be based on the UUT design or schematics. From Modular Automatic Test Equipment (MATE) program inputs, it was found that the UUT simulator should be oriented more toward ensuring that the V&V provides better checks on the TRD and on the test programs than toward minimizing the cost of developing test programs.
- The UUT simulator should be used as an orthogonal check on the validity and completeness of the testing as embodied in the TRD and in the test programs. Assessment of fault coverage completeness to minimize LRU down-time when in the field is a key payoff factor.
- The UUT simulator should be based on multiple levels of simulation and on either a software-only or an instrument bus

interface with the ATE system on which the test programs are executed. Solutions based on analog signal interfaces are not recommended.

- In using the UUT simulator as recommended above, the Test Package Set (TPS) development costs may actually increase. However, the resulting increase in test program quality (and consequently improved UUT uptime) should more than offset this increase.

- A UUT simulator is not a total or the only solution in supporting V&V and in decreasing test program development and maintenance costs. A real UUT must still be used during a part of the validation. Automatic test program generation, if realizable and cost-effective, reduces the need for a UUT simulator.

- Test program developers and validators must be convinced of the merits of a UUT simulator. A demonstration program addressing further issues and supporting the validation of the UUT simulator concept is necessary before large scale usage of UUT simulators.

- Additional UUT simulator cost-effectiveness issues must be resolved. It is not clear yet whether it is most cost-effective to develop a generic UUT simulator system, whether families of such systems should be developed, or whether it will suffice to simply develop a set of guidelines to support the construction of unique UUT simulators based on specific weapon program requirements.

SECTION 2

PROBLEM DEFINITION

This section presents a more detailed picture of the problem that the UUT simulator is addressing. Specific examples are used to demonstrate to the reader the reality of the problem. Experiences with a very low fidelity form of simulator are also described.

VALIDATION AND VERIFICATION (V&V)

Verification and validation of UUT test programs involve a complex set of disciplines. When the validator encounters a problem implementing a test, he must discriminate where the problem originates:

- Is it the test program software?
- Is the test program stepping outside the ATE capabilities?
- Is the ATE itself at fault? If so, is it by design deficiency or malfunction?
- Is the interface test adapter mating the UUT to the ATE at fault? If so, is it be design deficiency or malfunction?
- Is the UUT failing? If so, how can this be verified?
- Is the Test Requirement Document (TRD) for the UUT accurate, up-to-date, and correct?

Figure 1 depicts one version of an ATLAS test program development process. Though greatly affected by prior elements, V&V begins at the step "Validate Program with Adapter and UUT" and continues beyond acceptance of the first Test Package Set (TPS). The reason V&V actually continues beyond "sell off" is perceivable when considering multiple test systems deployment. Each ATE system must "play" with each of its TPSs to be operational. Experience at Warner Robins Air Logistic Command (WRALC) has shown that because of UUT variability, there is only 25 percent confidence that a TPS, when validated with only one UUT, will work with all UUTs. This confidence jumps to 75 percent for a TPS validated against two UUTs and 90 percent for three UUTs.

Typically, however, V&V is thought of as just through sell off of the first TPS. When used in this context, V&V usually accounts for about 40 percent of the TPS development process.

PROBLEM IS REAL AND COSTLY

To characterize the problem, let us look at an example derived from Honeywell experience. As an ATE supplier of the F-15 ADTS and F-15 TITE systems, Honeywell has total ATE and TPS development, integration, and V&V responsibility. This V&V activity has produced over 500 Avionics LRU and module TPSs during the last five years. Because each TPS effort involved usage of at least one UUT, at a few thousand dollars value each, many millions of dollars worth of UUTs have been tied up during this V&V effort. In the case of LRUs, the problem can sometimes be more

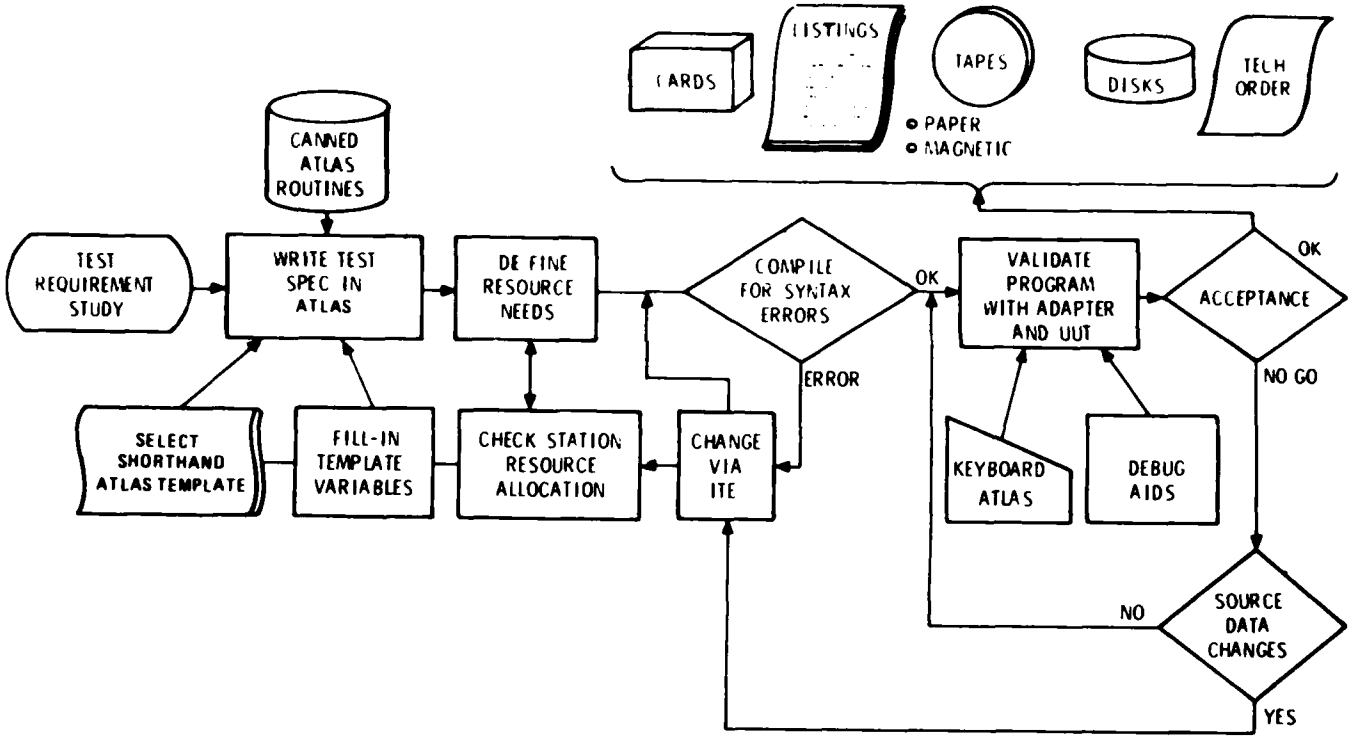


Figure 1. ATLAS Test Program Development Process

noticeable because the V&V cycle is usually longer and the LRU cycle is more costly. For example, on F-15 TITE, LRU No. 3 (Countermeasures Computer) required availability of three LRUs for one and one-half years during V&V.

UUT AVAILABILITY PROBLEM EXAMPLES

At Honeywell we have found that more typically LRU V&V requires an availability of one and one-half LRUs. The extra one-half LRU would generally be for card/plug-in module replacements for trouble-shooting and repair of the LRU in V&V. While this is possible with LRUs, it is not always practical, and it will not suit module level V&V at all.

These practical limitations on V&V activities were very clearly illustrated in the minutes of two recent F-15 TITE Technical Meetings:

- "There is a shortage of Band 1 UUTs. There is a total of four Band 1 UUTs; three of these four UUTs are defective. The only good UUT is being used in RF amplifier testing. Two of the other UUTs are being returned to St. Charles for repair. The other defective UUT will be kept at the site (V&V) for limited use during validation. The diodes required for repair of these UUTs have arrived from Northrop."
- "Verification of ATE Stations 10 through 15 at Honeywell, Minneapolis, presents no problems for the ICS UUTs; however, TITE does not have available enough EWWS and RWR UUTs for verification at Honeywell, Minneapolis, while maintaining UUTs at the St. Louis site for sustaining work."

UUT DAMAGE/REFURBISHMENT PROBLEMS

Going beyond availability of good UUTs, V&V faces the inevitable damage of a UUT during V&V. Typical causes for such UUT damage are as follows:

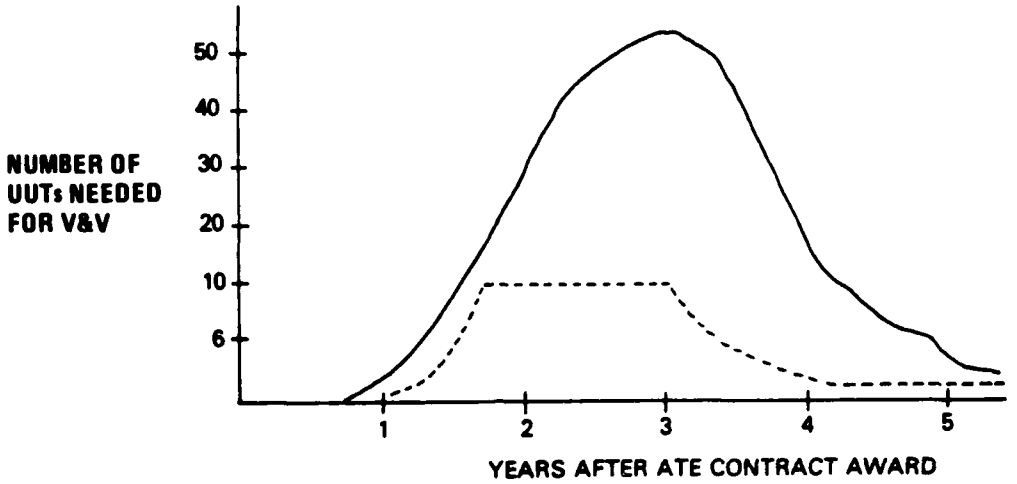
- Oscilloscope probe damage to Avionics module protective coating of components
- Inadvertent electrical damage to the UUT by grounding with probes
- Incorrect application of UUT power

While the first cause may be understandable in a validation environment and not too expensive to repair, the second and third causes can be very costly because damage to both the UUT and ATE can be extensive. But even if damage is minimal, a defective (damaged) UUT will usually shut down V&V effort because a second UUT is not available. Honeywell experience on F-15 shows the average time to repair a UUT is one to two weeks, and a minimum of 3 days is required. This discontinuity in the V&V effort has a very adverse effect on cost and schedules for ATE projects.

ELEMENTS OF A UUT SIMULATOR

Classes of UUTs

The typical requirements for UUT availability in a TPS development project are portrayed in Figure 2 (solid line). The dashed lines in this



—— Typical requirements for UUT availability in a TPS development project

--- Goal of reducing need for UUT

Figure 2. UUT Requirements at V&V Efforts

figure illustrate the goal of driving down the need for UUTs and reducing the time they are involved in V&V. If a universal UUT simulator were available that could stand up to V&V in place of any UUT, this could indeed drive the dashed line near zero. This problem is bounded somewhat by requiring coverage of only analog, digital, and hybrid UUTs. However, all classes of UUTs are encountered in TPS development and as such these "classes"--

- Analog
- Digital
- Hybrid
- High Power
- RF

- Microwave
- Video
- Pneumatic
- Inertial
- Communication

should also be considered as worthy targets of the solution.

The ATE Itself

The ATE where TPS development is conducted should also be considered an element in any simulation approach. Modularization of ATE promised by the MATE program could have a dramatic impact on a UUT simulator solution. If interfaces in MATE are standard and accessible, a simulator may be able to intercept native ATE functions rather than be a slave to the general ATE I/O ports.

Perhaps, though, the ideal simulator would not concern itself with the ATE, but rather conduct itself as the UUT would--responding to ATE stimuli with an expected UUT response. In this way, the UUT simulator could be used on any ATE, not just those with MATE-defined interfaces. Besides, it may not be practical to allow a V&V effort to have access to internal ATE functions.

Limited Data

Any UUT simulator must also address the problems of attaining information about the UUT. UUT Test Requirements Documents (TRD) misinformation has been proven, time and time again, to be a back breaker of TPS development programs. Acquiring information necessary for UUT simulation could be an equally frustrating experience. For a simulator to conduct itself as the UUT would perform, it may be necessary to obtain UUT design information as well as UUT performance information as contained in the TRD. This may be another practical limit to simulation, considering the consequences of vendor proprietary design information.

Consideration should be given to approaches to simulation that minimize the need for design information about the UUT. Perhaps the simulator should concern itself with only the UUT I/O interfaces—the signal types, ranges of stimulus and response constraints, data envelop formats, etc. Such information, because it may be more readily attainable and be fairly correct (because a companion LRU or module depends on this interface), may be a viable data base for construction of a UUT simulator.

Can the UUT Be Eliminated?

Unless one narrows and descopes the classes of UUTs and the types of tests validated on these UUTs (i.e., simple, moderate, or complex synchronized tests), it would be difficult, if not impossible, to totally eliminate the UUT from V&V. A UUT simulator approach that could eliminate the need for a UUT during 99 percent of the V&V effort, but was effective only for digital card UUTs, might be easy to achieve. But

another approach, which would eliminate the need for a UUT 80 to 90 percent of the time and which would cover a broader class of digital, analog, and hybrid LRUs or modules may, indeed, provide a better ratio of performance to development cost.

A relatively simple but effective simulation approach was recently developed by Honeywell for the F-16 Depot TPS development. For F-16 Depot, Honeywell was chosen to supply analog test stations. General Dynamics/Fort Worth (GD/FW) had TPS development responsibility. Because of schedule constraints, GD/FW needed a mechanism that would allow them to begin TPS development prior to delivery of analog test stations. As a result, Honeywell delivered the F-16 Depot Mini-Disc Expanded Memory System (UG26000GD01) in August 1978. As in many ATE program development stations, this system allowed test program generation to proceed in a very effective manner. But because this system, called Mini-DEMS, was implemented with a set of computer and peripheral devices identical to the analog test station, TPS V&V, as well as test system training capabilities, were added. This was accomplished by adding software and a circuit card to the computer I/O bus extension that would act as a null interface, "simulating" any analog test station ATE instrumentation. The Mini-DEMS has allowed GD/FW to actually execute the ATLAS test programs they develop. This Mini-DEMS has allowed the V&V process to proceed, eliminating not only the need for a UUT, but also the ATE itself and the interface test adapter. In addition, because the Mini-DEMS reacts just like the analog test station, it has served as a very effective ATE operation trainer also. Two Mini-DEMS were delivered to GD/FW, one in August 1978, the other in September 1978 (one will be delivered to USAF later).

TPS development and V&V have proceeded on a multishift basis at GD/FW since this delivery. Those test programs that had completed V&V testing on a Mini-DEMS were ready for final integration with the ATE which arrived in January 1979. Using this approach, 30 to 40 percent of the V&V effort was possible without the UUT.

SECTION 3

UUT SIMULATOR REQUIREMENTS

This chapter contains the definitions of the requirements which must be satisfied in order for the UUT simulator to realize the projected benefits. The first section contains a brief description of the methods used to collect the requirements. It shows that a comprehensive multi-faceted approach was used to arrive at the requirements. Reaching the same conclusions with different approaches lends credibility to the requirements.

The second section contains a definition of the baseline requirements for the UUT simulator. This set of requirements is fundamental to the operation of the simulator in the anticipated scenarios. If these requirements are not met, we project that the expected benefits of the UUT simulator will be lessened.

The third section contains a definition of the UUT simulator enhancements, prioritized according to their expected value. The features described here are expected to provide some payoff if included in the simulator, but they are "wants" rather than "musts." The prioritization was performed on a basis of payoff rather than on a basis of expected additional cost of the feature.

The remaining sections of this chapter contain additional documentation supporting the baseline requirements and enhancements. The topics discussed are requirements information from the literature, questionnaire results, and TRD/ATLAS program analyses.

REQUIREMENTS COLLECTION METHODS

A properly scoped activity to collect information, which impacts the requirements definition, is the key first step in arriving at a good UUT simulator definition. The collection must be performed from different angles of attack to ensure a good composite picture of what the problem is and what is needed. This was accomplished on our program by performing the following activities:

- A Literature Survey--A number of approaches was used to search for relevant information in the literature, which would help to clarify the nature and extent of the problem being addressed, which would describe relevant experience with simulation techniques or other methods for overcoming the problems, and which would possibly identify solutions that have already been used. During the literature search we did not concentrate as much on solutions as on trying to identify the problem. We were more interested during this stage of the program in information that would lead to requirements rather than solutions.

  The results of the literature survey are discussed in more detail in this section. These results and further observations from papers which we reviewed but did not document had a strong influence on the UUT simulator requirements defined in this chapter.

- A Survey Questionnaire--Based on preliminary discussions with test program developers and validators, a survey questionnaire was developed and distributed both within Honeywell and to outside experts. The results from the responses and in some cases

verbal discussions helped to formulate requirements constraints. By using the questionnaire we were able to find information which typically would not appear in formal papers in conferences.

- TRD/ATLAS Program Analysis—A representative set of test requirements documents and ATLAS test programs was chosen from the F-15 TITE program. These were then analyzed for complexity and for characteristics which would influence the requirements for a UUT simulator. The detailed results of this analysis are found in this section. Again, the UUT simulator requirements presented in this chapter reflect what was learned in the analysis.

When all the information had been collected, we looked at a number of alternative ways of coalescing it and presenting it. We finally decided to use the form of categorization of the requirements as shown in Table 1. This method allows a reasonably orthogonal presentation of the UUT simulator features and supports a structured presentation which will make it easier for the reader to keep the requirements separated. The following briefly discuss the categories which are used.

UUT Simulator Start Time (R.1)

This requirement has to do with the relative phasing of the UUT development, the development of the test program set for the UUT, and the development of the UUT simulator. In the ideal case, the UUT simulator could be started when the UUT is still in design. In some cases the UUT simulator must be

TABLE 1. UUT SIMULATOR REQUIREMENTS CATEGORIES

R. 1	UUT Simulator Start Time
R. 2	Modeling Levels
R. 3	Physical Requirements
R. 4	Time Requirements
R. 5	V&V Session Functionality
5. 1	Initialization
5. 2	Interactive Simulation Session
5. 3	Interactive Session Control
5. 4	Performance Constraints
5. 5	Simulation Session Preparation
5. 6	Post-Simulation Evaluation
R. 6	Simulation System Control Support

developed after the UUT has already been fully developed. It is still important to have adequate design and interface information to be able to develop an adequate simulator.

Modeling Levels (R. 2)

The level of modeling of the UUT determines the level of fidelity at which the UUT simulator mimics the actual UUT. At one extreme, digital and analog UUTs are simulated with precise circuit models. At the other extreme, only the interfaces which interact with the ATE equipment are sufficiently modeled to produce representative responses. The two extremes can also be considered analogous to models used in electronic maintenance training simulators:

- A "simulation based" maintenance trainer is one in which the interaction is based on simulation models of the device being trained. Internal states are maintained so that the simulation is reasonably realistic.
- A "procedural" trainer is one in which only the interface is simulated to a sufficient degree to allow a student to follow a sequence of steps in a Technical Order.

The "procedural" form of trainer corresponds to a UUT simulator based on interface characteristics only. This form of a simulation tends to be less expensive, but adequancy of fidelity is an issue.

For automatic test program verification there are additional modeling requirements beyond the UUT: the interface adapter, any operator interfaces (such as pots or switches), and the ATE equipment (depending on the simulation approach).

Physical Requirements (R. 3)

These requirements have to do with the required level of ruggedness of the equipment performing the UUT simulation, the location of the equipment relative to the ATE to which the UUT is connected, and the fidelity of the electrical signals at the interface. Physical portability is also a requirement to be considered in this category.

Time Requirements (R. 4)

There are a number of different aspects of time which may be required by the UUT simulator:

- Operation speed as compared to real time
- Block allocation time which may constrain the user access to the simulator
- Immediacy of access
- Response time to activities requested by the user
- Familiarization time for a user to become proficient in the use of the simulator

V&V Session Functionality (R. 5)

The requirements in this category have to do with the use of the UUT simulator during a session in which an automatic test program is being verified. To help partition the requirements into a more structured format, the following subsections are used:

- Initialization (R. 5. 1)
- Interactive Simulation Session (R. 5. 2)
- Interactive Session Control (R. 5. 3)
- Performance Constraints (R. 5. 4)
- Simulation Session Preparation (R. 5. 5)
- Post-Simulation Evaluation (R. 5. 6)

Each of these subsections is briefly discussed below.

Initialization (R. 5. 1)—Initialization includes activities required to start the system, to activate the desired UUT simulation model, to start the operator interface, and to activate auxiliary models such as the Interface Adapter model if present.

Interactive Simulation Session (R. 5. 2)—Requirements in this subsection are key in making the UUT simulator achieve the expected benefits. The user must be able to adequately control the interaction of the simulation with the ATE and the external environment. Furthermore, the simulation must provide him with feedback and must be adequately responsive so that he uses his time in debugging and verifying test programs efficiently. The requirements are further divided into the following categories:

Input Requirements (R. 5. 2. 1)—All inputs which must be sensed by the UUT simulator from the operator, from the ATE, and from the simulated environment are included. These inputs are simulations of equivalent inputs which must be sensed by the real UUT.

Output Requirements (R. 5. 2. 2)—All the outputs which must be generated by the UUT simulator are included. Also specified in this section are the means by which the outputs are calculated.

Simulation Status Outputs (R. 5. 2. 3)—In addition to the outputs which mimic the real UUT outputs, the UUT simulator can have additional internal states which can help in the process of debugging and verifying the test programs. This section contains the requirements for outputting such information.

Performance Monitoring (R. 5. 2. 4)—During the process of simulating a UUT being exercised by a specific test program, information can be collected on the performance of the test program and on the performance of the UUT. This information can then be used in evaluating how well the test program is exercising the UUT.

On-Line Performance Evaluation (R. 5. 2. 5)—Functional requirements in this category determine what performance information can be extracted while the simulation is still in progress. As a result of getting such information, the user makes changes in the test program or in the method of simulation to improve the performance indicators which he is watching.

Procedural Branching Control (R. 5. 2. 6)—An important aspect of a UUT simulator is the ability to force the test program into exercising all of the test branches. The requirements determine the different types of procedural control needed.

Patch and Retain Temporary Changes (R. 5. 2. 7)—During the debug phase of test program development it is necessary to allow the programmer to make incremental modifications so that he can gauge their effect on the simulation and on how the test program responds.

Interactive Session Control (R. 5. 3)—This set of requirements characterizes how the user can control the progress of his session on the UUT simulator. The requirements are further classified into the following categories:

Freeze/Run/Replay (R. 5. 3. 1)—Capabilities are specified here as to the extent of control of the simulation in terms of stopping it, allowing it to continue, and restarting it at a previous point.

Monitoring Level Control (R. 5. 3. 2)—The requirements in this section determine the amount of control over tracing of simulator variables.

Utility Functions (R. 5. 3. 3)—This section includes requirements for special utility functions which are needed to support control of the simulation.

UUT Simulator Messages (R. 5. 3. 4)—The requirements in this section include the types of messages which must be provided to indicate to the user that the simulator has encountered error conditions or that capacities of the system are close to being exceeded.

Performance Constraints (R. 5. 4)—This section contains requirements on the performance of the UUT simulator during a V&V Session. The two areas covered are performance, when used in a batch mode, and responsiveness, when used in an on-line mode.

Simulation Session Preparation (R. 5. 5)—Requirements in this category are concerned with the features in the UUT simulator, which are oriented to helping the user plan and prepare for a simulation session. The following are the subsections:

Function Selection (R. 5. 5. 1)—The requirements are included for selecting specific cases from a set of alternatives available. For example, selection of a specific UUT and selection of specific operating modes may have to be supported.

Malfunction Selection (R. 5. 5. 2)—The requirements in this section include the ability to force the simulator to mimic specified malfunctions which will, in turn, force the test program into specific procedures.

Procedure Selection (R. 5. 5. 3)—If response in the simulation can be determined by procedures, this section contains requirements on the selection and modification of these procedures.

Simulation Preparation/Selection (R. 5. 5. 4)—This section includes requirements on the ability to prepare specifics of the simulation such as amount of detail and values for specific parameters.

Trace Level Selection (R. 5. 5. 5)—The requirements included determine the ability to preset trace levels and detail before the simulation is started.

Post-Simulation Evaluation (R. 5. 6)—The requirements in this section include capabilities of the system to support evaluation of results after the simulation has been completed. The following are the subsections included:

Test Program Evaluation (R. 5. 6. 1)

Save Performance Reports (R. 5. 6. 2)

Display Specific Reports (R. 5. 6. 3)

Simulation System Control Support (R. 6)

Requirements in this section include those features of a computer system required to keep the system operational and to keep the system updated with the latest information on the simulation components. The following subsections are included:

Initialization (R. 6. 1)

Operational Assurance (R. 6. 2)

Subsystem Switching Control (R. 6. 3)

Off-Line Activity Control (R. 6. 4)

System Instrumentation (R. 6. 5)

System Recovery (R. 6. 6)

Configuration Management Support (R. 6. 7)

Performance Data Reduction Support (R. 6. 8)

Load Projection/Enhancement Support (R. 6. 9)

Data Base Support (R. 6. 10)

Off-Line Preparation/Maintenance (R. 6. 11)

A summary of the UUT simulator requirements results is presented in the matrix in Table 2. The requirements categories discussed above appear with two columns. The first column contains an "x" if the corresponding requirement is in the baseline. If there is no "x" in the first column, the feature is classified as an enhancement. In this case there is a number from 1 to 3 in column 2. A "3" indicates high value of the feature and a "1" indicates a low value. The individual requirements are further explained in the following two sections of this chapter.

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS

1. UUT Simulator Start Time		
At UUT Design		2
After UUT has been built	x	
2. Modeling Levels		
Circuit Level		
Analog		
Topology and Simple Models		2
Topology and Good Models		2
Good Models and Second Order Coupling Effects		1
Digital		
Register Transfer Model		2
Register Transfer Model and Timing Model		2
Detailed Circuit Model		1
Interface Level		
Recognize Stimuli and Generate Response (from bus)	x	
Recognize Stimuli and Generate Accurate Electrical Signals		1
Additional External Models		
Interface Adapter Model		
Topology	x	
Circuits		1
Operator Interface Model	x	
ATE Equipment Model		3

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

Modeling Language		
English Language Based (ATLAS-like)	x	
Graphical		2
Fault Simulation Capability		
Interface Level	x	
Functional Module Level		2
Circuit Level		1
3. Physical Requirements		
Hardware Reliability		
Militarized		1
Ruggedized		2
Commercial	x	
Location		
Adjacent ATE		3
Remote and Real Time Access		1
Remote Only		1
Physical Portability		
Movable among ATE Stations		3
Transportable to Different Sites		3
Electrical Signals		
Bypassed (via bus interface)	x	
Nominal Values (not accurate)		1
Accurate Values		2
Accurate Signals + noise effects		3

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

4. Time Requirements		
Real Time (UUT Operation)		
Full Real Time		1
Scaled, Reduced time		1
Asynchronous	x	
Block Time (How Quickly How Long)		
Continuous Usage		1
Allocatable by Block Time		3
Access Requirements		
On Demand		3
Flexible Schedule	x	
Fixed Schedule		1
Response Time (Validators Perspective)		
No Perceptible delays from real UUT operation		2
Up to 2 Min delays (or 5 sec/step)		3
Familiarization Time		
No Training Required (Built in Aids)		3
2 Day Training + Month Familiarization		1
About as long as learning ATE Use		2

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

5. V and V Session Functional Requirements			
5.1 Initialization			
System Initialization			
Display	x		
Panel Switches	x		
Restart at Saved State	x		
UUT Model Identification			
Verify Proper Model	x		
Provide Parameter input for model	x		
ATE Interface Model Identification			
Verify Proper Model	x		
Provide Parameter Input for Model	x		
Operator Interface			
Initialize Simulator Unique Functions (eg. Starting Collection)	x		
Select Operation Mode	x		
5.2 Interactive Simulation Session			
5.2.1 Input Requirements			
Input Sensing Requirements			
UUT Panel / Manual Inputs			
Soft Keys	x		
Test Points		3	
Dynamic (Manually Controlled) Model Parameter		2	

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

ATE Signal Routing	x	
Interface Signals		
ATE Inputs	x	
Recognize Valid Signal Gen. Command		2
UUT Environment Simulator Cooling Monitor		2
5.2.2 Output Requirements		
5.2.2.1 Calculate Outputs Based on		
ATE Inputs	x	
Operator Inputs on Simulated Panel/Controls	x	
Sequence of Input States		1
Functional Simulation Model (Digital)		1
Functional Simulation Model (Analog)		1
Logic Model (Digital)		2
Detailed Circuit Model		2
5.2.2.2 Calculate Responses based on Malfunction Simulation		
Single Malfunction	x	
Multiple Sequential Malfunctions		3
Multiple Concurrent Malfunctions		1

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

5.2.2.3 Generate Outputs		
UUT Simulated Indicators		
Alphanumeric Representation	x	
Graphic Representation		1
Outputs to ATE Interface		
Emit Realistic Responses	x	
Check validity of Measurement Command		2
5.2.3 Simulation Status Outputs		
5.2.3.1 Information Output		
Input State from ATE		2
Output State to ATE		2
UUT Indicator States		2
Selected Computed Values		2
Internal UUT States		1
History of States		3
5.2.3.2 Output Method		
Console CRT		3
Printer		3
On Mass Storage		3
5.2.4 Performance Monitoring		
Active Interface Pins		2
Signal Ranges on Interfaces		2
Closeness of signal limits to bounds limits		2

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

5.2.5 On-line Performance Evaluation			
Unused Interface Pins			2
Average Closeness of Signal to bounds Values			2
Fault Coverage			3
5.2.6 Procedural Branching Control			
Manual Control of Response to Force Test Branching	x		
Control of Branching via Stored Sequences	x		
Control of Branching via UUT Parameter Selection	x		
Control of Branching via Malfunction Selection	x		
5.2.7 Patch and Retain Temporary Changes			
Change Parameters	x		
Change UUT Response Procedure	x		
Change UUT Model Incrementally	x		
Substitute UUT Model Components			3
Store Model for Later Recall	x		
5.3 Interactive Session Control			

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

5.3.1 Freeze/Run/Replay		
Freeze State from Simulator Console	x	
Freeze State From ATE Command		1
Automatically Trigger Freeze on		
UUT Simulator Bounds Error		3
Expected Sigral Error		3
After Specified no. actions		3
Programmed Condition		3
Freeze and Restart		
After Fixed Delay		1
On Operator Initiation	x	
Save Frozen State on Command from Console	x	
Save Frozen State on Command from ATE		2
Restore Previous State	x	
5.3.2 Monitoring Level Control		
Trace Control		
Based on Console Level and Coded Threshold		2
Based on Programmed Conditions		1
5.3.3 Utility Functions		
Soft Keys for UUT Simulator Control	x	

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

5.3.4 UUT Simulator System Messages			
Simulator Error Conditions	x		
System Status		3	
System Error Conditions	x		
5.4 Performance Constraints			
5.4.1 Batch Performance			
Less than 1 hour turnaround		3	
5.4.2 Response Time Constraints			
Less than 2 seconds in Single Step Mode		3	
5.5 Simulation Session Preparation			
5.5.1 Function Selection			
Select Function from a Function Menu ..			
UUT to be Simulated	x		
Monitoring Modes	x		
Interaction Modes	x		
5.5.2 Malfunction Selection			
Select Specific Malfunction Procedures	x		
Select Parameters Inducing Malfunctions	x		
5.5.3 Procedure Selection			
Select Response Procedures	x		
Build New Response Procedures and Chain	x		

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

5.5.4 Simulation Preparation/Selection			
Select Simulation Detail Level	x		
Select Specific Simulation	x		
Select Simulation Parameters	x		
5.5.5 Trace Level Selection			
Based on Preset Level		3	
Based on Programmed Conditions		2	
5.6 Post-Simulation Evaluation			
5.6.1 Test Program Evaluation			
Generate Statistics on			
Coverage(Steps, Pin Usage etc.)		3	
Tolerance Limits		2	
Error Summaries		3	
Generated Simulated UUT Statistics		3	
Generate Simulator System Statistics		2	
5.6.2 Save Performance Reports			
On Mass-storage		3	
Changes from stored reports		2	
5.6.3 Display Specific Reports			
On Display		3	
On Printer		3	
Compared to Saved Reports		2	

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (continued)

6. Simulation System Control Support		
6.1 Initialization	x	
6.2 Operational Assurance	x	
6.3 Subsystem Switching Control		1
6.4 Off-line Activity Control		2
6.5 System Instrumentation		1
6.6 System Recovery		1
6.7 Configuration Management Support	x	
6.8 Performance Data Reduction Support		1
6.9 Load Projection/Enhancement Support		1
6.10 Data Base Support		
UUT Models	x	
ID Characteristics		3
ATE Interface Configuration		2
6.11 Off-Line Preparation/Maintenance		
6.11.1 Model and Procedure Preparation Support		
Interactive	x	
English-Language Based	x	

TABLE 2. A SUMMARY OF UUT SIMULATOR REQUIREMENTS RESULTS (concluded)

6.11.2 Model and Procedure Code Modifications		
Incremental Modifications	x	
6.11.2 Interface Model Modifications	x	
6.11.4 UUT Simulator Verification		
Syntax Checks		3
Inconsistency Analysis		1

BASELINE REQUIREMENTS DEFINITION

This section contains a discussion of the set of UUT simulator requirements which are in the baseline. This set of requirements was determined to be needed in order to realize the benefits expected of a UUT simulator for use in support of the V&V of test programs.

The requirements discussions follow the structure as defined in Section 3, Requirements Collection Methods. Only those items in Table 2 which contain an "x" in column 1 are discussed here.

UUT Simulator Start Time (R.1)

After the UUT Has Been Built--Based on comments from papers in the literature and from responses to our questionnaire, it can only be assumed that the UUT simulator is started after the UUT has already been built. It would be desirable to start the development of the UUT simulator during the UUT development to use as much of the available design information as possible, particularly when the test programs are developed by an organization other than the developers of the UUT; this has not been done. One of the major problems in test program development is that not enough information is generally available on the UUT to expeditiously develop the test programs. Usually a sequence of iterations of information collection is needed before the test program development can be completed.

To maximize the utility of the simulator as an independent check of the test programs during V&V, it is highly desirable that the simulator simulate the UUT as closely as possible. The UUT simulator should

therefore be based on UUT design information such as that contained in a Part II specification and a schematic. For an independent check, it is particularly important that the UUT simulator not be based on the requirements in the TRD or the code in the ATLAS test program developed to test the UUT.

Development of the UUT simulator independently of the test program development can serve as a secondary check on the adequacy of the information about the UUT. The developers can cross-check their understanding of the UUT when the test programs are finally run against the UUT simulator.

Modeling Levels (R.2)

Interface Level--

Recognize Stimuli and Generate Response (From Bus)--This level of modeling would recognize that a specific stimuli is being applied (via the ATE equipment command and the routing which is in effect). The recognition assumes the ATE is based on a bus structure architecture and that the UUT simulator can intercept and respond on the bus to mimic the stimuli generators and instruments.

Depending on the ATE approach, this level of modeling appears to provide most of the expected benefits and is reasonably feasible.

Additional External Models--

Interface Adapter Model: Topology--The topology of the interface adapter must be capable of being simulated to support checking of proper signal routing during the simulation.

Operator Interface Model--Any interface between the UUT and the operator (such as switches and indicators) which the operator must observe or set, must also be capable of being simulated. See the requirement below on "soft keys" ("Utility Functions").

Modeling Language--

English Language Based (ATLAS-like)--If any form of UUT simulator modeling language is required (even if only very simple), it must be English Language based and similar in intent to ATLAS. The primary objective is to make the resulting models easy to read and easy to alter for an appropriately skilled model developer.

Fault Simulation Level--

Interface Level--Faults must be capable of being simulated in order to force the test programs into their no-go and diagnostic paths. However, as a baseline requirement it is necessary that only the faults be simulated at the interface level with interface characteristics which trigger the fault detection tests in the test programs.

Physical Requirements (R.3)

Hardware Reliability--

Commercial--The hardware which is used to implement the UUT simulator should have about the same reliability requirements as the ATE system. High quality commercial equipment is adequate.

Electrical Signals--

Bypassed (via bus interface)--Comments from the people surveyed and comments from papers in the literature place little credibility in achieving adequate quality electrical signals without excessive cost. Also, there is a feeling that the effort required to verify that the simulator is functioning properly at the signal level may be too costly in terms of the developers' time.

Therefore, the baseline should have, at most, the signals represented digitally over the buses connecting the ATE central computer, the ATE instruments, stimuli, and the Interface Adapter to the UUT simulator If a software solution to the UUT simulator is shown to be feasible, even this bus interface may not be necessary.

Time Requirements (R.4)

Real Time (UUT Operation)--

Asynchronous--Neither real time or even scaled real time are baseline requirements for the UUT simulator. The combination of the UUT simulator and the ATE system can operate synchronously at their natural speed.

Access Requirements--

Flexible Schedule--The simulator should be available to the test program developers and validators on a flexible schedule basis. On-demand scheduling is highly desirable, but not a baseline requirement. Putting the simulator on a fixed schedule is too constraining.

V&V Session Functionality (R.5)

Initialization (R.5.1)--

System Initialization--

- Display--All displays must be capable of being initialized.
- Panel Switches--All panel switches must be capable of being put into an initial condition either automatically or via instructions to the operator.
- Restart at Saved State--It must be possible to restart a simulation (including all displays and external switches) from a saved state.

UUT Model Identification--

- Verify Proper Model--It must be possible to verify that the proper model of the desired UUT is going to be used.
- Provide Parameter Input for Model--During initialization it must be possible to provide for key model parameter inputs, which allows the user to make the next simulation run unique from previous runs.

ATE Interface Model Identification--

- Verify Proper Model--If the ATE Interface is to be simulated, it must also be possible to verify that the proper model is going to be used.
- Provide Parameter Input for Model--It must be possible to provide for key model parameter inputs for the ATE model, if the ATE is to be modeled.

Operator Interface--

- Initialize Simulator Unique Functions--It must be possible to initialize any unique function which the UUT simulator has that involves interfacing with the operator. Presetting switchings and making simulated jumper connections are examples.
- Select Operation Mode--A number of alternative operator interface modes may be available (e.g., totally manual, semi-automatic, or totally automatic). During initialization it must be possible to choose the proper mode for the session.

Interactive Simulation Session (R. 5. 2)--

Input Requirements (R. 5.2 . 1)--

- *Input Sensing Requirements*
  --UUT Panel/Manual Inputs--The UUT simulator must be capable of sensing "soft keys" which mimic manual inputs and actions which would be performed on the real UUT.
  --ATE Signal Routing--Depending on the simulator, it may be necessary to simulate the Interface Adapter and the signal routing which the ATE is forcing.
  --Interface Signals--The UUT simulator must be capable of handling representations of input signals from ATE stimuli generators appropriate for the recommended level of simulation.

Output Requirements (R. 5. 2. 2)--

- Calculate Outputs Based on (R. 5. 2. 2. 1)
  --ATE Inputs--The simulator must be capable of calculating outputs based on input conditions.
  --Operator Inputs on Simulated Panels/Controls--It must also be possible to generate outputs based on the state of simulated panels and controls which may have to be manipulated by the operator because of operator action requests in the test program.

- Calculate Responses Based on Single Malfunction Simulation (R. 5. 2. 2. 2)

  --Single Malfunction--It must be possible to have the UUT simulator mimic the occurrence of a single malfunction which results in the appropriate symptom in the simulator responses.

- Generate Outputs (R. 5. 2. 2. 3)

  --UUT Simulated Indicators

  Alphanumeric Representation--If the UUT has indicators which the operator is required to read or react to, the UUT simulator must be capable of outputting the states, at least on an alphanumeric display. It is not considered necessary to output the information graphically or on a realistic panel. It is assumed that the test program developer or validator is adequately trained in making the alphanumeric to indicator status translation.

  --Outputs to ATE Interface

  Emit Realistic Responses--It must be possible for the simulator to emit realistic responses appropriate to the level of simulation fidelity recommended.

Procedural Branching Control (R. 5. 2. 6)--It must be possible to control the branching in test programs via the following methods:

- Manual Control of Responses--It must be possible to allow the user of the UUT simulator to manually control the simulator response in order to force the program to take

a desired branch. This capability is analogous to making a manual alteration in a real UUT to force test program branching.

- Control of Branching via Stored Sequences--It must be possible to force the test program being exercised to take a sequence of actions in the UUT simulator. These stored sequences would typically be used to revalidate a sequence of tests which had been operating correctly before a modification to the test program was made. The capability is required to ensure that test program modifications do not have unexpected and unintended side effects.

- Control Branching via UUT Parameter Selection--It must be possible to allow parameters in the UUT simulator to control branching in the test program. These parameters would be set at initialization time or dynamically during a simulation.

- Control of Branching via Malfunction Selection--It must be possible to control the branching in a test program by selecting specific simulated malfunctions in the UUT.

Patch and Retain Temporary Changes (R. 5. 2. 7)--It must be possible to make temporary patches to the UUT simulator in order to make it respond differently. The reason for this may be that a more detailed model is required, that the original response was not correct, or that a trial is being made to see how the test program will respond. After it is verified

that temporary change is to be made permanent, it must be possible to retain the change in permanent storage. The following areas of change must be supported:

- Change Parameters of the UUT Simulator
- Change UUT Response Procedure
- Change UUT Model Incrementally
- Store Model for Later Recall

Interactive Session Control (R. 5. 3)--

Freeze/Run/Replay (R. 5. 3. 1)--

- Freeze State From Simulator Console--It must be possible to freeze the state of the UUT simulator and of the ATE from the simulator console. The purpose of this freeze is to allow the user to examine the state of the UUT simulator and, if desired, to allow him to change or restart the simulation.
- Freeze and Restart
  --On Operator Initiation--Control of restart must be exercisable from the UUT simulator control console.
- Save Frozen State on Command from Console--A command must be supported to save the state of the UUT simulator after a freeze. The purpose of saving the state is to allow a later restart at the same state. This feature is required to minimize the time required to get back to a point of interest in the test program debugging or verification.

- Restore Previous State--The restore command which is the inverse of the save command must be supported from the simulator console.

Utility Functions (R. 5. 3. 3)--

- Soft Keys for UUT Simulator Control--If the UUT to be simulated has switches or other manual inputs, they must be supported by "soft keys" on the UUT simulator console.

UUT Simulator System Messages (R. 5. 3. 4)--

- Simulator and System Error Conditions--The simulator must be capable of generating messages for the user in case of simulator error conditions or in case of error conditions in the hardware or software on which the UUT simulator is implemented.

Simulation Session Preparation (R. 5. 5)--

Function Selection (R. 5. 5. 1)--

- Select Function from a Function Menu--During simulation preparation it must be possible to select functions from a function menu. The following are specific areas of functionalities which must be selectable:

  --UUT To Be Simulated--It is assumed more than one UUT may be supported by one simulator system and that multiple versions of a specific UUT simulator may be present.

--Monitoring Modes--Determining the amount of tracing and checking (if implemented).

--Interaction Modes--Determining how the simulation is to be run (from completely interative to batch oriented).

Malfunction Selection (R. 5. 5. 2)--

- Select Specific Malfunction Procedures and Select Parameters Inducing Malfunctions--During a simulation session preparation, it must be possible to select specific malfunction procedures which will force test program branching. It must also be possible to select UUT parameters which will force malfunction symptoms during the simulation session.

Procedure Selection (R. 5. 5. 3)--

- Select Response Procedures--It is assumed that a comprehensive set of procedures will be accumulated to simulate the UUTs responses to test program actions. During a session preparation, it must be possible to select sets of these response procedures to specify how the UUT simulator will respond during the simulation session.
- Build New Response Procedures and Chain--It must also be possible to build new response procedures for the UUT simulation. Incrementally building up such response procedures must be one way of gradually building a simulator for a specific UUT.

Simulation Preparation/Selection (R. 5. 5. 4)--

- Select Simulation Detail Level, Select Specific Simulation, and Select Simulation Parameters--If various levels of detail are possible in a UUT simulation, it must be possible to select the desired sets to be used in the session. Specific simulation models used previously must also be selectable. Parameters which control the simulation must also be selectable.

Simulation System Control Support (R. 6)

Initialization (R. 6. 1)--The UUT simulator system must be capable of being initialized in a simple manner after power is turned on. The normal computer operating system procedures for initialization are assumed to be adequate.

Operational Assurance (R. 6. 2)--The simulation system must have resident in its file system a set of programs which, when executed, will ensure that the system is ready and performing adequately to run the simulator.

Configuration Management Support (R. 6. 7)--The simulator system must contain a file system and operating system which are adequate for configuration management support of all of the software components which make up the UUT simulator system and specific UUT simulators.

Data Base Support (R. 6. 10)--

UUT Models--The simulator system must be capable of supporting the UUT model data base at minimum.

ENHANCEMENTS

This section contains descriptions of desirable UUT simulator enhancements beyond those required in the baseline. The list of enhancements is included in Table 2 as explained in Section 3, Requirements Collection Methods. The descriptions below provide more details about the enhancements and the rationale for the weighting which was assigned to them. Only those items in Table 2 which are classified as enhancements are described below.

UUT Simulator Start Time (R. 1)

At UUT Design--The ideal time to start the definition of a UUT simulator would be late in the design of the UUT when most of the design decisions have been frozen. It is at this time that the design decisions are still fresh in the minds of the developers and much of this information could be put into an appropriate form for the UUT simulator. If circuit models were used in the development of the UUT, the information from these models could be transformed into information needed in a UUT simulator to support test program development.

Note that the information in the models of the UUT needed for design verification is different from the information needed for test program

development. But it can be conjectured that the information needed for test program development could be derived from that in the design verification model.

A value of 2 was assigned to this feature to reflect the anticipated difficulty and credibility in being able to start on the UUT simulator so early in the design of a UUT. Also, if too much detail is used, it may get in the way of supporting test program debugging and verification. Keeping up with UUT design changes is also a potential problem.

Modeling Levels (R. 2)

Circuit Levels--

Analog--

- Topology and Simple Models--Modeling at this level, if possible, could still be relatively fast. But reservations about the adequacy and payoff versus the anticipated effort were expressed. A value of 2 was assigned because of these reservations.
- Topology and Good Models--Though the modeling would be better than above, the resulting speed is expected to be lower. Furthermore, the same reservations about adequacy and payoff still hold. A value of 2 was assigned.

- Good Models and Second Order Coupling Effects--Even with good models, second-order coupling effects would still be needed to model some types of analog failures. The added complexity and reduced simulation speed are not considered effective for a UUT simulator to support test program development. A value lower than 1 should have been assigned, but because of the convention, a 1 had to be used.

Digital--

- Register Transfer Model--A UUT simulator embodying a register transfer model of the internal UUT functionality is considered more useful than detailed models for the analog case. Such a model should be particularly useful for UUTs which are too complex for automatic test program generation application.

  The utility of a register transfer model to support test program debugging and verification is still questionable. It is possible to perform such modeling independent of the ATE system so that when the test vectors or stimuli patterns are applied, the results need not be questioned as much (as is the case if the test vectors are generated by ATPG). For these reasons a weight of 2 was assigned.

- Register Transfer Model and Timing Model--This is a higher fidelity version of the case discussed above. The

resulting model would be slower but more useful. But the same relative utility question can be raised, and consequently a weight of 2 was assigned.

- Detailed Circuit Model--Models such as those used for automatic test program generation are too detailed for test program development general usage. A value of 1 was assigned.

Interface Level--

Recognize Stimuli and Generate Accurate Electrical Signals--For this case the recognition would depend on recognizing signals across UUT pins and generating actual electrical responses. To achieve adequate fidelity, the user would have to be concerned about many signal details which are judged as getting in the way of doing test program debugging and verification. A value of 1 was assigned.

Additional External Models--

Interface Adapter Models--

- Circuits--The additional complexity of this level of IA modeling is judged to get in the way of doing test program debug and verification. A value of 1 was assigned.

ATE Equipment Model--An equipment model is particularly useful if a software UUT simulator solution is to be used or if the ATE system is based on a bus architecture and the UUT simulator is interfaced to the ATE on this bus. A value of 3 was assigned.

Modeling Language--

Graphical--A graphical communication of the UUT internal model and interface model would be useful in some cases to minimize development time. But because of the extra effort needed to use it and because it would not be as universally applicable as an English Language based system, a value of 2 was assigned.

Fault Simulation Capability--

Functional Model Level--Introducing simulated faults modeled by functional modules appears desirable. The value above what can be done beyond that at the Interface Level for supporting test program debugging and verification is an issue. A value of 2 was assigned.

Circuit Level--Supporting faults simulation for the full UUT at the circuit level appears too detailed to be effective in debugging and verifying test programs. A value of 1 was assigned. Circuit level simulation may be useful in a multi-level simulator.

Physical Requirements (R. 3)

Hardware Reliability--

Militarized--Using militarized components and standards is not warranted. A value of 1 was assigned.

Ruggedized--Even ruggedized requirements are too stringent to be requirements for the UUT simulator implementation. A value of 2 was assigned.

Location--

Adjacent ATE--It is most desirable that the UUT simulator be capable of being placed adjacent to the ATE system with which it is interacting. The final form of the UUT simulator will determine how valuable the relative location is. A value of 3 was assigned.

Remote and Real-Time Access--If access to the simulator must be close to real time, having the UUT simulator implemented on a remote computer is not desirable. A value of 1 was assigned.

Remote Only--If the simulator is strictly software, it could be implemented on a remote computer. Feedback from our survey indicated that this is not very desirable. A value of 1 was assigned.

Physical Portability--

Movable Among ATE Stations--It is desirable that the simulator be capable of being rolled to allow positioning beside different ATE stations. A value of 3 was assigned.

Transportable to Different Sites--Ease of transporting the simulator or making it easily accessible to different sites facilitates support of TPS development by organizations different from the developers of the UUT. A value of 3 was assigned.

Electrical Signals--If electrical signals are to be generated (particularly in the analog case), they must be accurate and realistic to be useful to support debug and verification of test programs. Therefore, a simulator which generates only nominal values was assigned a value of 1; one that produces accurate signals, a value of 2; and one that produces accurate signals and noise effects, a value of 3. Cost-effectiveness analysis is expected to show that the latter, particularly, is not feasible.

Time Requirements (R.4)

Real Time (UUT Operation)--Full real time or even scaled time are not considered high needs for test program debug and verification. A value of 1 was assigned to both.

Block Time--Allocation of the UUT simulator by block time appears adequate and was assigned a value of 3. Having the simulator available

on a continuous basis (for example, by dedicating a system or by time-shared support) would be nice, but it was not judged a strong need. A value of 1 was assigned.

Access Requirements--Having the simulator available on demand would be useful, particularly during test program debug, but it is not a requirement. A value of 3 was assigned. A fixed schedule is less desirable because it may cause developers' time to be wasted. Therefore, a value of 1 was assigned.

Response Time--Having no delay as compared to the real ATE and real UUT would, of course, be the ideal case. But developers are conditioned to expect delays and to plan for them. Therefore, requiring no delay was assigned only a value of 2.

To prevent wasting developers' time, some limit, such as two minutes, should be a maximum wait goal for normal types of operations. If the developer is interacting directly with the UUT simulator (such as in stepping through a program) no more than a five second delay should have to be tolerated.

Familiarization Time--The following values were assigned:

No Training Required (Built in Aids): 3

2 Day Training + Month Familiarization: 1

About as long as learning ATE Use: 2

V&V Session Functionality (R. 5)

Interactive Simulation Session (R. 5. 2)-- See Table 2.

Interactive Session Control (R. 5. 3)--See Table 2.

Performance Constraints (R. 5. 4)--

Batch Performance (R. 5. 4. 1)--If the simulator is to be used in batch mode, it is desirable to provide adequate turnaround to allow a developer to be concurrently analyzing two or three segments of his test program. In this way he can analyze one set of results while waiting for the others to get back. A turnaround time of one hour appears a reasonable goal.

Response Time Constraints (R. 5. 4. 2)--If the developer is interacting with the UUT simulator system (such as editing or monitoring a trace output), a response goal of two seconds should be targeted. If the system becomes much slower, the developer is wasting time and may become frustrated.

Post-Simulation Evaluation (R. 5. 6)--See Table 2.

Simulation System Control Support (R. 6)

Subsystem Switching Control (R. 6. 3)--Systems that require high uptimes may have to be supported by dynamic configuration control so that a faulty component can be switched out and a replacement can be switched in.

Availability requirements for the UUT simulator are not high enough to warrant such complexities. A value of 1 was assigned to this feature.

Off-Line Activity Control (R. 6. 4)--It is sometimes desirable to perform background support activities at the same time that a simulation is in progress. Though this is desirable for the UUT simulator, it is not essential. A value of 2 was assigned.

System Instrumentation (R. 6. 5)--To help ensure that the system is performing optimally, it is desirable to have a good complement of system instrumentation. These features are also useful if it is determined that the performance of the simulator is not good enough and selective components must be improved. The instrumentation then supports finding where the performance problems are and what improvements can be made.

System Recovery (R. 6. 6)--It is desirable to have automatic recovery after a power failure or a system fault. Because of the save and restore features already in the baseline, additional system recovery features are less useful.

Performance Data Reduction Support (R. 6. 8)--It is desirable to have system support which will help to reduce the system performance data and present it in summary form.

Load Projection/Enhancement Support (R. 6. 9)--If the UUT simulator becomes heavily used and if multiple users are supported, features which help projection of loading on the simulator and which help to evaluate

potential enhancements (such as added memory or added disc space) will be useful in getting maximum benefit out of the UUT simulator.

Off-Line Preparation/Maintenance (R.6.11)—

UUT Simulator Verification (R.6.11.4)—One of the problems which must be addressed is verifying that the UUT simulator performs as required. The generic support modules which are used for all implemented simulators must be carefully verified, as any other software is. However, the specific UUT simulator models also require verification support. A first-cut capability would be extensive syntax checks of the descriptions of the simulators. A more sophisticated capability would support tools to anyalyze potential inconsistencies.

REQUIREMENTS INFORMATION FROM LITERATURE

A reasonably extensive search was performed in the public literature which may have a bearing on the problem we are addressing. The references listed at the end of this report are a result of this search and secondary leads from papers which were already found. We had found additional references, but because the papers were not conveniently accessible and in most cases of marginal relevance, they are not listed.

The papers were reviewed and classified according to relevance on different topics. The results of this classification are presented in Appendix A. The topics which were used for classification were as follows:

- UUT Reference
- ID (IA) Reference
- TRD Reference
- Analog UUT
- Digital UUT
- Test Program Development Costs Information
- TPS Development Difficult Information
- UUT Difficulties
- ID (IA) Difficulties
- ATE Difficulties
- Payoff Information
- Development Tools Descriptions

Note that not all references were classified. Some references were obtained after the classification was performed.

During Phase I of the program these references were viewed primarily in context of what they say about the problem which we are addressing. Little attention was paid to information on potential alternative solutions or techniques useful for a UUT simulator. The papers will be reviewed again in Phase II to extract this type of information.

The insights in some of the papers influenced our thinking in the development of the UUT simulator baseline requirements and in the prioritization of the enhancements.

QUESTIONNAIRE RESULTS

The questionnaire prepared for the survey attempted to answer many questions without introducing the author's bias. This approach resulted in a number of questions that were open to interpretation but unburdened by specific solution biases. One of the goals of the questionnaire was to get a measure of the effect the UUT simulator might have on various phases of V&V. Many of the questions attempted to identify areas of the simulator's functionality on which to focus increased emphasis. The other major goal of the questionnaire was to provide a forum for the expression of opinion and suggestions for improvement to the V&V process.

The questionnaire approach was taken both to get the broadest possible coverage and to provide those surveyed with the most time to consider the questions.

The questionnaire and summary responses to the questions follow.

As a Test Program Validator

1. Did you have your test program or Interconnect Device (ID) development delayed by the unavailability of a UUT?
   yes____ no____

If yes, what percentage of the programs that you worked on had this problem?

6 yes
1 no

Responses ranged from "a small number" to "100% of the test programming or ID development efforts were delayed by the unavailability of UUTs."

2. Did you have a "Golden" UUT exhibit a failure?

yes _____ no _____

6 yes
1 no

Probably fewer than 1%

3. Did you lack sufficient time on the ATE to debug your test program and ID?

yes _____ no _____

If yes, how would you have relieved this problem?

5 yes
2 no

Provide "more stations" interleave ID and test program design, use a "simulator", "do more work off station."

4. Did you find a significant percentage of your debug time devoted to debugging your ID?

yes _____ no _____

If yes, what percentage? _____

3 yes
4 no

Percentage of debug time ranged from 10-50%.

5. Did you have trouble getting what you felt was the necessary source data to develop your test program?

yes _____ no _____

If yes, describe the problems and what you could have done to alleviate them.

5 yes | Solution: "Do the best you can with what you have."
2 no |

6. Did you find that the Test Requirements Document (TRD) did not adequately test the device?

yes _____ no _____

Was the testing described incomplete?

5 yes | Noncommittal responses generally left judgment of the adequacy of the TRD to the end user.

7. Did you spend more time analyzing the UUT than would have been necessary if you would have had a simulator?

yes _____ no _____

2 yes | The remaining three responses expressed concern that
2 no | they might be debugging the simulator.

8. Did you have trouble preserving the integrity of low level signals?

yes _____ no _____

If yes, describe the problem.

4 yes | "Station, UUT, and ID Noise" were identified as sources
3 no | of noise problems.

10. Did you have grounding problems?

yes _____ no _____

If yes, describe them.

4 yes
3 no | Grounding problems were attributed to an ID system ground inadequacy.

11. Did you have trouble understanding the TRD?

yes _____ no _____

If yes, why?

4 yes
3 no | Different writing styles and vague requirements were the main problems in understanding a TRD.

What Were the Problems and How Would You Rank Them in Severity?

A. Test Program Development Problems

1. What type of UUTs (analog, digital, etc.) required the most time to debug and/or validate? How much time was required for each?

   A four to one ratio of analog to digital test program development time was given, qualified by other responses indicating that it depended on the number of tests in the TRD.

2. What do you believe the reasons are for one type of device requiring more debug/development/validation time than another?

   ATG was cited in two responses as the reason why digital was easier. Other causes were length of TRD and complexity of the test.

3. How do you work around these problems?

   Responses were generally "to dig into" the problem. A more thoughtful response was to create special software to permit more direct access to test program statements needing debug.

4. What tools or aids would you like to have had to help solve these problems?

   Suggested tools included a system trace that would keep track of the station's status (what stimuli had been applied, relays closed, etc.) as the test progressed, aids to explain ATLAS fields, analog ATG, ATE simulator, bus and logic analyzers, user programmable "soft" keys, a UUT simulator, and an automated ID checker.

5. What percentage of your test program debug time was spent on the following:

   a. checking the logical flow of the software?

   b. verifying signal levels, tolerances, and waveshapes?

   c. verifying the adequacy of the test requirements document?

   d. verifying the operation of the UUT as assumed by your test program?

   e. resolving grounding and noise problems? Please describe these problems and aids that you believe could relieve these problems.

   f. resolving problems with low-level signals?

The following six responses were received for the distribution of debug time:

a.	checking S/W logical flow	30%	1%	50%	10%	60%	5%
b.	verifying signals	70%	75%	10%	30%	10%	50%
c.	verifying TRD	--	1%	5%	20%	15%	10%
d.	verifying UUT operation	--	13%	20%	20%	10%	30%
e., f.	Resolving grounding and noise problems	--	10%	15%	20%	5%	5%

B. ID Associated Problems

1. What type of problems did you experience in checking out your IDs? How much time was spent on each?

   Misinterpretation of TRD requirements and production errors were most commonly cited ID problems.

2. Which of these were the most time consuming?

   Misinterpretation of TRD requirements was cited as the most time-consuming problem.

3. What tools or aids would have relieved these problems?

   "An automatic ID checker" was the only tool cited. Other suggested improvements were procedural in nature.

C. Station Associated Problems

1. How significant was the availability of time on the ATE station?

   All the responses indicated that availability of station time was very significant.

2. How much station time was spent debugging IDs?

   Station time spent on debugging IDs was "very little," "15% of validation," "50% too much" and "70% of the validation effort."

3. How would it have been useful to be able to run a test program on a software model of the ATE station?

   "Very important!", "only to verify...correct branching," and an extensive affirmative answer based on the use of a "null interface" simulation that Honeywell had provided for his use.

4. How significant would the operator interface presentation have been for a model of the ATE station?

   The operator interface presentation "must be identical," is "very significant."

D. UUT Associated Problems

1. How often and how significant a problem was the lack of a UUT or the failure of a "Golden" UUT?

   "Less than 10% of the total time" seems to be the consensus; one response said late UUTs were a "significant" problem.

2. What percentage of the lost time in test program development was due to incomplete source data?

"Very little" to "50%;" consensus seems to be around 10%.

E. TRD Associated Problems

1. What percentage of lost time was due to inadequate test requirements definition?

Responses ranged from 10% to 50%--clustered the 50% end.

2. Are there any tools or aids that would have helped relieve these problems?

More attention devoted to TRDs including one suggestion that they be written in ATLAS.

How Would You Validate a TPS if its UUT Was Difficult to Obtain or its Availability Was Limited?

A. What could be done to improve current validation efforts?

1. What engineering data is required to validate (and prove the validation of) a TPS?

Validate the TRD against the program.

2. What are the key parameters to be validated in a TPS?

Software sequencing, signal transfer by the test program through the station and ID to the UUT, TRD and test program tests match.

3. Which of these parameters could be validated without a test station?

   The interfaces (UUT, ID, station) and software logic flow could be validated without a test station.

4. How valuable would a program which would develop an interface cross-reference list from a finished program be?

   "Great".

5. Would a program which scanned your test program (or TRD) to match its resources requirements to a given ATE system be of value to you?

   yes _____ no _____

   If not, why not?

   Unanimously yes with the one cautionary response that the resource allocator not detract from available station time.

B. If a UUT simulator were available, what attributes do you think it should have?

1. What human factors should be considered?

   a. Access via a general purpose terminal?

      yes _____ no _____

   b. A specialized modeling/simulation keyboard?

      yes _____ no _____

c. Librarian facilities (storage and retrieval of models and submodels for repeated use or incorporation into other models)?

yes _____ no _____

d. Ability to incorporate user-defined algorithms in the simulation process?

yes _____ no _____

e. Other considerations?

All of the sub-questions were ignored. Written comments expressed the concern that the simulator might require as much validation as the UUT. A more positive comment was that the simulator should not require all of the station hardware and should produce a high-speed printout of all station actions.

2. How much training would you consider reasonable to learn to use the simulator? How should it be taught?

The same or less than 1/2 the time now required for UUT test design.

3. At what level of detail (complexity) should the simulator model the UUT?

The single definitive response indicated that a simulator that addressed the I/O pins with internal functional models of circuit components would be the appropriate level of detail. A macro-level model was thought to be of minimum value.

4. What level of interface fidelity should the simulator have?

   a. Complete electrical and mechanical fidelity?

   b. Ability to support check out of the interconnect device?

   c. An interface which would simulate the loading, noise, etc. of the ATE on which the test program was to run?

   d. An interface in which pre-defined (possibly by the user) responses are supplied in response to test program measurement statements?

   Of two definitive responses, they agreed that complete electrical fidelity was required, and to questions b, c, and d they responded "more work, not less."

   e. A simulation in which the simulator analyzed the test program and exercised its various diagnostic paths?

      "Definitely, with trace capability" and a printout of station actions on a high-speed printer.

   f. Can you identify a preferred interface via an analogy with an existing simulation system with which you are familiar? (If the system identified is not widely available and in the public domain, we would appreciate any conveniently available documentation.)

      The "null interface" system provided to the F-15 site was cited as a successful V&V aid.

5. Do you consider it necessary to be able to control the level of detail at which the simulation operates?

   yes _____ no _____

   If not, skip to question number 6.

   "The amount of data produced during a given test run should be optional."

6. How accessible must the simulator be?

   a. Constantly available?

   b. Access scheduled in blocks of time?

   c. Should it be interactive?

   One response indicated it should be constantly available; the other said four hours/day; both agreed it should be interactive.

7. Describe the manner in which you would see a UUT simulator used.

   "This device should be a station simulator, not a UUT simulator. All of the adapter and UUT characteristics should be provided by the respective designers to be used by the simulator."

8. What can be done to ensure that the UUT simulator is cost-effective in supporting test program verification and validation?

   One response indicated that a UUT simulator would never be cost effective; the second suggested that the simulator be "a combination program development and station simulator."

TRD AND ATLAS PROGRAM ANALYSES

To present a more in-depth feel of test program development and verification, a small representative set of UUT test programs was analyzed. This section gives a brief description of the analysis and the results.

The test programs analyzed are items from the F-15 TITE program. These program sets were developed by Honeywell, based on formal TRDs from the customer.

TRD Analysis

The particular UUT TRDs selected from TITE were the following:

- LVPS — Low Voltage Power Supply
- RF AMP — RF Amplified
- FL — Fault Locator
- LRU-3 — Countermeasures Receiver

The following is a brief description of these units.

- LVPS

  The LVPS is a module contained in the band 1 and band 2 oscillators.* The low voltage power supply converts 115 VAC 400 Hz 3 phase aircraft power to the DC voltages required in the band 1 and band 2 oscillators.

---

*Band 1 and band 2 oscillators are externally cooled LRUs.

- RF AMP

  Band 1

  The band 1 RF AMP is an externally cooled unit. It receives jamming parameters from the band 1 oscillator. These parameters are modulated, amplified, and then transmitted.

  Band 2

  The band 2 RF AMP is also an externally cooled unit. The band 2 RF AMP receives jamming parameters from the band 2 oscillator; these parameters are modulated, amplified, and then transmitted.

- FL

  The fault locator is a module contained in the band 1 and band 2 oscillators. This module continuously or on demand tests the band 1 or band 2 osciallator and the RF AMP. This fault locator module performs a built-in test (BIT) for the controlled oscillator and the RF AMP LRUs.

- LRU-3

  The low band receiver/processor detects and processes low band transmissions and analyzes them in conjunction with high band transmissions to determine all threat parameters. It uses the computer within the low band receiver processor.

The purpose of the test requirements document analysis was to determine the types and complexity of the tests and the structure of the TRD. Three specific features of the TRD tests were tabulated.

1) Simple tests such as voltage, current, or resistance measurements were counted. These types of tests are of interest because of their easy implementation on the UUT simulator.

2) Non-simple tests were rated on a scale from 1 to 10 (the most complex was rated as a 10). This was done with the intention of having the scale correspond to the level of difficulty of writing a section of code for a specific test.

3) The number of high-power tests was also determined. During these tests the UUT must sink or source more than 1 watt of power. This is of importance in regard to a simulator, because the simulator would need special generators and heavy duty internal circuitry to support high power tests.

The structure of the TRD was examined by looking at the following characteristics:

1) The percentage of the total tests used for fault isolation was obtained. This is of significance because the simulator would be helpful in checking fault-isolation paths.

2) The number of entry points into the fault isolation section of the TRD was also determined. This is important because the use of a simulator would aid in checking these paths.

3) The number of branches in the normal test part of the TRD was also determined. This is useful in establishing how much a simulator would benefit branch checking.

The complexity of the stimuli and response was estimated separately, and they are often different. Stimuli of a constant voltage or resistance were not classified as complex stimuli; likewise, a constant voltage or resistance measurement was not classified as a complex response. The basic idea behind the complexity levels is that they represent how difficult the signals would be to produce. What follows is a brief definition of the levels by example.

1. Stimuli
   a. A step function with no rise time or duration specified

   Response

   a. Measurement of two discrete levels, one before the step function and one after

      Each of the specific levels has an upper and lower limit.

2. Stimuli
   a. A pulse of specified duration
   b. A step function with step occurring at a specific time with respect to the test

   Response

   a. Delay measurement of the response pulse to the stimuli pulse

3. Stimuli
   a. Many simple or level one tests
   b. Ramp function with a specified rise time

c. Simple pulse trains

d. A feedback loop where the stimuli voltage is changed until the response voltage changes to a different level

Then the threshold stimuli voltage is recorded as the response.

Response

a. The signal pulse will remain less than a certain value upon application of the stimuli (i.e., surges caused by application of prime power),

b. And those listed in stimuli category.

4. Stimuli

a. Moderately long pulse trains

Response

a. Several delay tests

b. Moderately long pulse trains

5. Stimuli

a. Long pulse trains

Response

a. Long pulse trains

6. Stimuli

a. Complex pulse trains

Response

a. Complex pulse trains

7. Stimuli

a. No stimuli found that have reached levels of 7 or above

Response

a. Complex waveforms with rise and fall times, pulse width limits, and lower and upper limits on voltage at quiescent levels specified

8. Stimuli

a. None

Response

a. Several responses with complex waveforms

9. Stimuli

a. None

Response

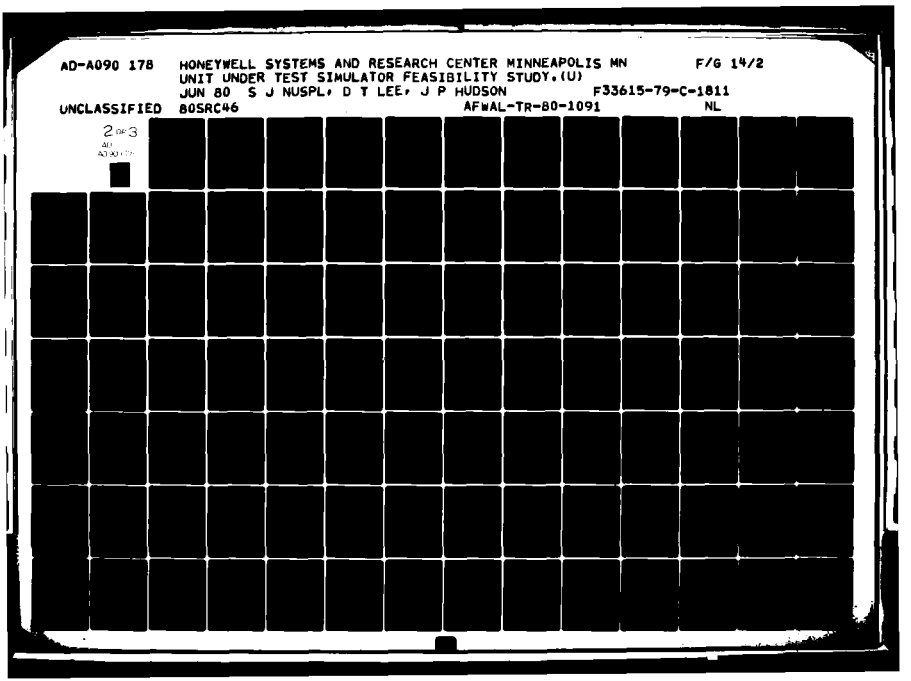
a. A pulse that must fit inside an envelope
This would give the waveform upper and lower limits on the period, rise and fall times, and voltage at all points on the waveform.

The TRD contains a description of the terminals that are on the UUT. Input, output, and test connections were counted and recorded on the TRD analysis table (Table 3).

TABLE 3. TRD ANALYSIS RESULTS

TRD	LVPS 094 007326	RF AMP 094 007327	FL 094 007325	LRU 2*
UUT Interface				
Number Connections				
Total	62	147	210	-
In	17	24	81	-
Out	36	19	76	-
Test	35	88	60	-
TRD Structure				
Number Tests				
Total	181	406	431	528
Normal	62	116	96	349
Fault Isolation	119	290	335	179
Maintenance Actions				
Normal Termination	1	4	2	2
Fault Isolation	49	92	97	0
Number Branches	54	93	94	-
Entry Points	27	33	49	-
Types of Test				
Resistance	25	6	72	55
Voltage, Current, Power	44	103	70	301
Complex Stimuli				
Number Steps	38	50	73	-
Average Complexity	1.8	1.7	2.2	-
Complexity Measure	68	85	158	-
Complex Responses				
Number Steps	28	72	84	153
Average Complexity	1.6	2.3	2.7	1.7
Complexity Measure	44	167	228	256
High Power Tests				
UUT Sink	81	177	150	-
UUT Source	81	3	0	-

* Data from LRU 2 was taken from a TRD Summary.


AD-A090 178 HONEYWELL SYSTEMS AND RESEARCH CENTER MINNEAPOLIS MN F/G 14/2
UNIT UNDER TEST SIMULATOR FEASIBILITY STUDY.(U)
JUN 80 S J NUSPL, D T LEE, J P HUDSON F33615-79-C-1811
UNCLASSIFIED 80SRC46 AFWAL-TR-80-1091 NL
2 OF 3
AD A090178

The number of tests in both normal test and fault isolation sections was tabulated. This portrays the length of the TRD.

A maintenance action is usually some physical action performed on the UUT or the ATE. The maintenance actions in both the normal test and the fault isolation sections were counted. Because this represents something a simulator would have difficulty performing, this is relevant material to consider.

The number of branches out of the normal test section was counted. Entry points into the fault isolation section were also counted. This information is useful in determining the logic flow of the TRD.

Resistance measurements and voltage and current measurements were counted. These are categorized as simple tests. Tests that were more complex than simple tests were rated according to the scale as discussed previously in this section. The number of complex steps for both stimuli and response was recorded. The complexity levels of all the steps for stimuli and response were summed and this information was recorded as the complexity measure. This was done separately for stimuli and response. Then the average complexity for stimuli and response was determined. This information was found by dividing the complexity measure by the number of steps. Once again this was done for both stimuli and response.

High-power tests that cause the UUT to sink or source high power are important. A simulator design would be altered significantly by the presence of many high-power tests. Therefore, the number of tests where the UUT must sink or source high power was recorded.

The following are the conclusions from the TRD analysis activity:

- The average complexity levels of the response were higher than those of the stimuli. This implies that a simulator would be different from an ATE. The simulator would have more sophisticated signal sources and generators than an ATE, but it would not require the sensitive sensors and measurement devices that an ATE has.
- Approximately 2/3 of the TRD was found to be composed of simple tests. A simulator that would handle just simple tests might prove to be cost-effective because of this finding.
- In the TRDs analyzed, roughly 2/3 of the total number of tests are used for fault isolation. This portrays the importance of having a simulator capable of checking the numerous branches that occur in fault isolation.
- Assumptions can be made relating the physical size of the TRD to its complexity levels and total number of tests.
    1. Complexity levels of the response were found to correspond to the thickness of the TRD. The thicker TRDs were found to have higher response complexity levels.
    2. A correlation between the thickness of the TRD and the total number of tests was found. The number of tests in a TRD increased with increasing TRD thickness.

ATLAS Program Analysis

Selected ATLAS test programs from the F-15 TITE project were analyzed. The analysis was performed with the intention of observing qualities that would be useful in establishing the baseline requirements for the UUT simulator. The analysis consisted of looking at the following characteristics of a test sequence:

1. The logic flow of selected tests was plotted. This was done to determine the importance of having a simulator capable of forcing control into the various branches of the test program.

2. Occurrences of operator messages and operator actions were tabulated. The operator is essentially another interface between the ATE and the UUT. If the operator must perform a substantial amount of communication between the ATE and the UUT, that would require a simulator to have more sophisticated operator I/O capabilities.

3. For a given test the number of statements was counted. This number was then compared to the complexity level established in the TRD analysis. This would provide information on how accurate the TRD stimulus and response complexity analysis was. Test programs that correspond to some of the analyzed TRDs were used for test program analysis. This was done to allow comparison between results of TRD analysis and the test program analysis. Also, because familiarity with a group of UUTs was obtained during the TRD analysis, the test program analysis was aided by choosing test programs for the same UUTs. The specific TITE LRUs tests which were analyzed can be found in Table 4.

TABLE 4. ATLAS TEST PROGRAM LOGIC FLOW DIAGRAMS

FL	LVPS	LRU-3	RF AMP
TP 47	TP 48	T4 T909	TP 83
TP 32		T4 T1926.20	TP 56
TP 56		T2 T9 (465-467)*	TP 38
TP 17		T2 T1	
TP 13		T3 T802	
TD 22.4		T4 T947	
TP 71		T2 T9 (310)*	
TP 23.2		T1 T12	
TP 37		T3 T883	
TP 39			

Tests in the test program are usually organized and identified in the same way as they are in the TRD. The test identifiers are TP, which is short for test performance (also known as normal test), T for test, and TD, which stands for test diagnostics or fault isolation. The FL, LVPS, and and RF amp tests are identified in their test program by the same identification given to the corresponding test in the TRD. Tests in the LRU-3 have a more complicated test identification code than was given to them in the TRD. This is due to the enormous size of the test program. The test program had to be divided into five segments to be stored on the disk file. The format used to identify a test is of the form TX TY (NNN), where TX is the segment indicator, and TY is the test identification taken from the TRD. If (NNN) is also present, it is a further breakdown of test TX TY. This breakdown, NNN, along with TX is not used to identify a test in the TRD; only TY is used for identification.

Appendix B contains specific examples of stimuli and response classifications. The examples make references to specific UUT pins, signal values, and measurement units. This was done to allow the reader to visualize the amount of information in the TRD.

Appendix C contains flow diagrams for some of the test program sections. The diagrams are classified by the acronym for the LRU and by the test number. In some cases the complexity of the stimuli and responses as determined from the TRD is also included. The straight-line code segments have been abbreviated, and a short word is given to present a rough indication of what the test program is doing. The important information to be conveyed in the flow diagrams is the amount and structure of the branching.

Results from the logic flow diagrams show that the tests are usually shallow, serial code. Several other more specific details about program organization and structure were observed.

1. No-goes are usually directed to fault isolation, or occasionally they are tested further to determine which entry points into fault isolation is correct.

2. Subroutines are seldom called during a normal test. In fault isolation, subroutines are used to obtain setup procedures or operator messages and manual intervention.

3. There was found to be a high level of correspondence between the TRD test number and the actual code sequence in the test program. Tests occurred in the same order in the test program as they did in the TRD.

The number of statements for a test was found to agree well with the complexity levels for the stimulus and response established in the TRD analysis.

Operator messages and manual interventions occurred on 5 to 10% of the normal tests, but appeared much more frequently in fault isolation. It was difficult to determine an exact figure because of the use of subroutines to perform operator messages and actions. An estimated 50 to 75% of the tests in fault isolation performed one or more operator message or action.

SECTION 4

SOLUTION ALTERNATIVES

This chapter presents a bottom-up view of the alternate solutions with which a UUT simulator could be realized. The presentation is structured to proceed from the applicable techniques and technologies, through the trade-offs, and into the recommended solution. The following summarize this progression:

- Simulation Techniques: This part of the chapter presents summaries of simulation techniques documented in the literature, which are potentially applicable to a UUT simulator.
- Applicable Technologies: A list of technologies and their applicability and implication on a UUT simulator are presented. The objective is to constrain the set of solutions considered for a UUT simulator to the technologies which will be available and proven by the time of implementation of the UUT simulator.
- Solution Components: Based on the requirements in Section 3, the simulation techniques discussed above and the technology constraints, specific UUT simulator components are identified. The solutions are then classified into a smaller set to permit them to be used in a trade-off.
- Trade-off Descriptions: This section of the chapter contains a discussion of the trade-off process and results in choosing a specific solution as the recommended UUT simulator approach

for subsequent refinement. A discussion is also included on the sensitivity of the choice to some of the key factors entering the trade-off.

SIMULATION TECHNIQUES

Computer simulation of electrical circuits has become a very important tool in the area of computer-aided design (CAD), automatic test equipment, and test program generation. Although each of these application areas may require some form of additional features such as fault insertion and fault isolation, the basic simulation capability required remains very much similar. In this section, the state of the art in a circuit simulation system is reviewed and assessed.

Simulation techniques to be used in the UUT simulator were found from documentation of existing techniques and from extensions derived by considering specific characteristics of the application of a UUT simulator oriented primarily to support TPS verification and validation.

The approach to describing these techniques here includes the following:

- Describe a set of categories which will help to differentiate the simulation techniques.
- Present more detailed descriptions of some of the more promising techniques in each category.
- Attach summaries of the references from the literature.

- Discuss the feasibility and appropriateness of considering specific techniques for use in a UUT simulator oriented to support TPS verification and validation.

In general, the simulation techniques currently used can be divided into five categories:

- Circuit Level Analog Simulation
- Digital Circuit Gate-Level Simulation
- Functional Simulation
- Interface-Level Simulation
- Mixed-Level Simulation

A survey of these techniques was conducted and is reported in the following subsections.

Circuit Level Analog Simulation

In this approach, the simulation is performed down to the individual circuit element component levels, such as resistor, capacitor, and transistor. As such, it can provide very accurate simulation with regard to signal amplitudes and timing analysis. This is particularly useful in performing circuit analysis in circuit design. Both analog as well as digital circuits can be simulated at this level. However, the number of computations involved could be overwhelming because elaborate circuit modeling algorithms have to be implemented. This technique is best suited for analog circuits. For large complex circuits, such as digital LSICs and VLSICs, the computation can get out of hand rapidly. In those cases, simulation programs at a higher level may be more efficient.

Currently there are more than 30 simulation packages that are available for circuit simulation at this level. Some of them are specialized packages that are used for special applications only. Together they can provide features such as statistical or tolerance analysis, sensitivity, and stability analysis, as well as DC, AC, and transient analysis. Some of the more prominent simulation systems that have been developed are shown in Tables 5 through 8, together with their characteristics and capabilities. The contents of these tables were extracted from "Computer-Aided Design," IEEE Spectrum, October 1975. To provide a better insight into the nature of a circuit level analog simulation system, one of the program packages is examined in more detail—the SPICE 2 system.

SPICE 2 is a general-purpose digital computer program to simulate the electrical performance of semiconductor circuits. It was developed by Dr. Lawrence Nagel of the Electronics Research Lab., University of California, Berkeley.

The program can be used to simulate most electronic circuits to determine the quiescient operating point, the time-domain response, or the small-signal frequency-domain response of the circuits. It provides for 12 types of built-in device models for the common circuit components. These 12 circuit elements include the following:

- Linear Elements
  Resistor
  Capacitor
  Inductor
  Coupled Inductors

TABLE 5. MAJOR COMPUTER-AIDED CIRCUIT DESIGN PROGRAMS

Program Name	Type of Analysis Linear	Nonlinear	Approx. No. of Statements (x 1000)	Core Requirements (k bytes)	Time In Use (Years)	Program Availability
ADA 74	x	x	4	64	14.5	1
ANP 3	x		4	80-135	8	2
ASTAP	x	x	60	200	10.5	1
BELAC	x		10	120	11.5	2
CIRCUS 2	x	x	18	180, 400, 600	9.5	2
COMPACT	x		2.5	96	9	1
CORNAP	x		3	80-110	12.5	2
ECAP 2	x	x	25	200-300	9	1
EXHOPT	x		1.5	250	8	2
IMAG 2	x	x	10	100	9.5	1
ISPICE	x	x	25	330	7	1
ITRAC 3	x	x	7	120	10	1
LISA	x		17	140	11.5	2
MARTHA	x			32	10	1
MOFRAN	x		4	156	6.5	2
NAP 2	x	x	6.5	104, 250	7	2
NET 2	x	x		70	11	3
NICAP		x	10	300-350	10	4
PTNA	x		1.5	64	8.5	2
SCAP	x		1.4	12	7	2
SCEPTRE	x	x	30	228	7	4
SLIC	x	x	10	200	8	2
SNAP	x		1.5	26	10	2
SPICE 1	x	x	10	184	8	2
SPICE 2	x	x	13	135 up	5.5	2
SUPER SCEPTRE	x	x	40	238	7	2
SYSCAP 2	x	x	20	155, 60	6.5	1

LEGEND: Commercially available: 1 Classified: 3
Handling charge: 2 No Charge: 4

TABLE 6. PROGRAM FEATURES

Program Name	Program Features
ADA 74	Optimized for most efficient use of CUP time and memory
ANP 3	Excellent teaching program written in FORTRAN and ALGOL
ASTAP	Statistical transient analysis
BELAC	Available on time-share, batch, and remote batch
CIRCUS 2	Convolution integral modeling of linear systems, third generation features
COMPACT	Microwave design and optimization, using (gradient) search: coarse (vicinity) search is optional, available on time-share
CORNAP	State equation oriented, written in ANSI FORTRAN 4
ECAP 2	Nesting of models and subcircuits, recursive modeling, capability for rerun with modified topology, interactive
EXHOPT	Use of Hessian Matrix, least square approximation
IMAG 2	Third generation features
ISPICE	Completely interactive or batch(at user's option)
ITRAC 3	Interactive
LISA	Provides poles and zeros and block diagram
MARTHA	Suits microwave and low frequency, built-in transistor and microwave models, and response functions
MOFRAN	Recursive use of subnetwork in ANSI FORTRAN 4
NAP 2	Third generation features, well-suited for stiff systems
NET 2	Hierarchy of subcircuits, up to 16 levels, built-in transistor and diode models, automatic parameter for modeling from curve
NICAP	Built-in models and a model library
PTNA	Educational aid for modern network analysis studies, completely topological approach
SCAP	Response evaluated by loopless code
SCEPTRE	Runs controls in transient analysis: start, stop, step, size, number of points
SLIC	Poles and zeros calculations
SNAP	Symbolic network function
SPICE 1	Built-in models for bipolar junction transistors, diode, junction FET, MOSFET
SPICE 2	Built-in models for bipolar junction transistor, diode, junction FET, MOSFET, dynamic memory
SUPER SCEPTRE	Sceptre features plus mechanical and control systems, digital and nonlinear mechanical models
SYSCAP	Circuit failure simulation and stress analysis

TABLE 7. ANALYSIS DETAILS

Program Name	AC	DC	Sensitivity	Worst Case	Monte Carlo	Time	Optimization	Fourier Analysis	FFT	Elements		Input Details				
										Active	Passive	Equations	Library Elements	Tables	Transmission Lines	Arbitrary Input Signals
ADA 74	x	x	x				x					x	x			
ANP 3	x	x	x	x		x				x	x	x		x		x
ASTAP	x	x			x	x				x	x	x	x	x		x
BELAC	x	x	x	x	x	x	x			x	x	x	x	x		
CIRCUS 2		x				x				x	x	x	x	x		x
COMPACT	x		x		x		x			x	x		x	x	x	
CORNAP	x					x				x	x					
ECAP 2		x				x				x	x	x	x	x		x
EXHOPT	x						x			x	x					
IMAG 2	x	x					x			x	x	x	x	x		x
ISPICE	x	x	x	x		x		x		x	x	x	x	x		x
ITRAC 3	x	x	x	x	x	x				x	x	x	x	x	x	x
LISA	x	x	x			x				x	x					x
MARTHA	x	x						x		x	x	x	x	x	x	x
MOFRAN	x									x	x					
NAP 2	x	x	x	x		x	x	x		x	x	x	x	x		x
NET 2	x	x			x	x	x			x	x	x	x	x	x	x
NICAP		x				x			x	x	x		x	x		x
PTNA	x	x									x					
SCAP	x	x								x	x				x	
SCEPTRE		x				x			x	x	x	x	x	x		
SLIC	x	x	x			x	x			x	x			x	x	x
SNAP										x	x					
SPICE 1	x	x	x			x		x		x	x		x			x
SPICE 2	x	x	x			x		x		x	x					x
SUPER SCEPTRE	x	x	x	x	x	x	x			x	x	x	x	x	x	x
SYSCAP	x	x	x	x	x	x		x		x	x	x	x			x

TABLE 8. OUTPUT DETAILS

Program Name	Element Volts	Node Volts	Element Circuit	Voltage Differences	Poles Zeros	Transfer Function	Gain	Driving Point Function	Group Delay	Phase	User Specified	Laplace Equations	List	Printer Plots	Histograms	Scatter Plots	Envelop Plots
ADA 74											x		x	x			
ANP 3					x	x	x	x	x	x			x	x			
ASTAP	x	x	x	x	x	x	x	x			x		x	x	x	x	x
BELAC	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
CIRCUS 2	x	x	x	x							x		x	x			
COMPACT						x	x	x		x	x		x	x			
CORNAP	x	x	x	x	x	x	x	x	x	x			x	x			
ECAP 2	x	x	x	x							x		x	x			
EXHOPT	x	x	x	x			x	x		x			x				
IMAG 2	x	x		x		x	x			x	x		x	x	x		
ISPICE	x	x	x	x		x	x			x	x		x	x			
ITRAC 3	x	x	x								x		x	x			
LISA	x	x	x	x	x	x	x	x	x		x		x	x			
MARTHA	x	x	x	x		x	x	x		x	x		x	x			
MOFRAN	x	x	x	x		x	x	x		x			x	x			
NAP 2	x	x	x	x		x	x	x		x	x		x	x	x		
NET 2	x	x	x			x	x	x		x	x		x	x	x		
NICAP	x	x	x				x						x	x			
PTNA	x	x	x			x	x				x		x	x			
SCAP	x	x	x	x									x	x			
SCEPTRE	x		x								x	x	x	x			
SLIC		x	x		x	x	x	x		x			x	x			
SNAP		x		x		x		x					x	x			
SPICE 1		x	x	x		x	x	x		x			x	x			
SPICE 2		x	x	x		x	x	x		x			x	x			
SUPER SCEPTRE	x	x	x	x									x	x			
SYSCAP	x	x	x	x			x			x	x		x	x	x		

Independent Voltage Source

Independent Current Source

Linear Voltage-Controlled Current Source

- Nonlinear Elements

  Nonlinear Voltage-Controlled Current Source

  Diode

  Bipolar Junction Transistor (BJT)

  Junction Field-Effect Transistor (JFET)

  Insulated-Gate Field-Effect Transistor (MOSFET)

Each model in turn contains many parameters which are used to reflect the individual device characteristics and the technologies used in the chip manufacturing. These parameter values should be updated periodically to reflect the current state of IC technology. As emerging technologies such as VMOS and SOS are adopted, entirely new model types will have to be incorporated into the package.

For performing the circuit analysis, the program uses a modified nodal analysis method which generates a well-conditioned system of equations that can be solved with a minimal amount of computational effort. The program can handle linear as well as nonlinear networks. By using the nodal analysis method, the system of algebraic-differential equations describing the complete circuit can be reduced to one of the three types of systems:

- System of Linear Equations
- System of Nonlinear Equations
- System of Ordinary Differential Equations

Of the many numerical analysis algorithms available for each type of the system, SPICE 2 uses a sparse matrix technique, a modified Newton-Raphson algorithm (the simple-limiting method of Colon), and the trapezoidal rule integration algorithm, respectively, to solve the three types of systems. A modified Gear's algorithm is also optionally available to solve the third type of systems.

In all, a combination of ten analysis types are available in the SPICE 2 program to perform the simulation of an electronic circuit. They include the following:

- DC Analysis

  DC Operating Point
  Linearized Device Model Parameterization
  Small-Signal Transfer Function
  DC Transfer Curves
  Small-Signal Sensitivities

- AC Analysis

  Small-Signal Frequency-Domain Response
  Noise Analysis
  Distortion Analysis

- Transient Analysis

  Time-Domain Response
  Fourier Analysis

The SPICE 2 system has approximately 15,000 source statements written in FORTRAN IV to run on the CDC 6400 computer. However, the program has also been adapted for use on the IBM and Honeywell computer systems.

Although this circuit simulator system is considered to be one of the industry standards, it has several shortcomings that need to be improved. Additional capabilities that would be desirable include:

- Model flexibility
- User specified functions
- Worst-case analysis
- Monte Carlo (statistical) analysis
- Radiation effect analysis

As mentioned before, most of the simulation systems described so far have been developed for circuit analysis in design automation. For direct application in a UUT simulator modifications and additional features will be required.

Note also that simulation of a discrete analog circuit can always be carried out, using circuit breadboards and analog computers. For integrated circuits, the presence of parasitic components makes this traditional method of measuring electrical characteristics of a circuit unacceptable.

Digital Circuit Gate-Level Simulation

For a digital circuit, a commonly used technique is to model the actual circuit as a connection of gates (AND, NAND, OR, NOR) and perform the simulation at this gate component level. This technique typically can provide accurate simulation to a detailed circuit level with respect to logic states and timing analysis. Thus, this technique is often used in digital test program generation systems where detailed simulation capability is required.

Most simulators model at least three logic states (logic 0, logic 1, unknown x) and use a unit delay timing model. For certain applications a higher accuracy is needed, requiring more logic states (such as high impedance $\not{z}$, rising edge $\uparrow$, fall edge $\downarrow$, etc.) or better timing accuracy.

Systems that utilize this simulation technique include:

D4-LASAR	(Teradyne)
CAFIG	(Bendix Corporation)
FAIRTEST	(Boeing Computer Services)
FANSIM	(GTE Laboratories)

In the D4-LASAR system, the only building block used is a NAND gate; i.e., all circuit elements are represented by NAND gates only. With this modeling technique, it is algorithmically simple to implement fault simulation and the path sensitization method in vector generation. In addition, it enables the internal operation of an IC to be evaluated at the detailed NAND level during fault simulation.

For the other three systems, instead of using the NAND function only, they allow a gate to be any of several Boolean functions with multi-inputs and single output. They also enjoy the similar advantages of algorithmic simplicity and detailed fault analysis.

The complexity of a circuit is generally measured in terms of "equivalent gate counts;" that is, a circuit is broken down into the basic building gates, such as NAND, NOR, and OR gates, and the count of all of these gates then determines the size and complexity of the circuit.

A circuit is represented as a logic diagram showing the interconnection of ICs. The coding of such a logic diagram can be fairly simple, requiring mainly specifications of the IC types and the interconnection of various IC and edge connector pins. Other systems such as D4-LASAR, however, would require a good understanding of digital logic and the simulation system in order to correctly model a circuit. This requirement is due to circuit conditions which cannot be modeled by the system and which call for a "workaround" technique, substituting the original circuit with special circuit elements.

A brief description of these four systems which use the gate-level simulation technique is given below:

D4-LASAR is one of the most commonly used automatic test generation systems in the industry. It was first developed in 1968. The basic principle behind D4-LASAR is for all circuits to be modeled in terms of NAND gates. The reasoning is that it is more efficient to run a system optimized for one device type rather than for one which has special codes

for many different types of devices. Also it is argued that a detailed NAND model of an IC is the only effective way to test out the IC's internal functions, because it allows for the analysis of stuck faults at the gate rather than at the IC level, and that a worst-case timing analysis is better carried out with a NAND gate model.

The main drawback of this system is that it is computationally inefficient. Besides, the user skill level must be relatively high in order to recognize the need for special models in many cases.

The D4-LASAR system is maintained by Teradyne, Inc.

CAFIG is a simple simulation system developed by Bendix Corporation mainly for in-house use. As such, it is not meant to serve as a general-purpose test programming system. Some of the restrictions on the system's capability are as follows:

- 1000 gate limit in circuit complexity
- No bus pin concept
- Manual wired nodes
- No one shot
- Slow fault simulation
- No interactive faulting and display capability

The FAIRTEST system is maintained by Boeing Computer Services which obtained the software from Fairchild Systems Technology.

The FAIRTEST system consists basically of three major portions: the good circuit simulator (FAIRSIM), the stimuli generator (FAIRGEN), and the fault simulator. The system is rated as an excellent good-circuit simulation capability. In particular, the simulation command language is very powerful, including conditional language commands. The good-circuit simulation can be carried out over three states with assignable delays (0 through 64) for transitions high and transitions low. There are altogether 12 primitives which can be used in modeling a circuit for FAIRSIM.

The fault simulator can simulate 255 faults in parallel. Faults included in the analysis are stuck faults, both external and internal to an IC, and shorts between adjacent IC pins.

FANSIM was developed by GTE Laboratories for use within GTE for design simulation, test verification, and automatic test generation of digital circuits. The system is well-known for its concurrent fault simulation.

The technical features in the good-circuit simulator include a nanosecond delay (up to 16,000) simulation capability in which two delays are assigned to each output of an element. The simulator also uses an inertial delay model for ICs to filter out multiple transitions which occur within the delay time of a device. Large pulses are flagged via special messages.

Using the concurrent fault simulation technique which GTE developed, the system is quite computationally efficient. To overcome the large memory requirements inherent in this approach, a hardware-based virtual memory system is used.

The IC and circuit modeling is generally convenient except for bus-structured circuits because there is no concept of an IC bus pin.

The characteristics and capabilities of these four systems can be summarized in Tables 9 through 14.

In general, the gate-level simulation approach seems to have the following advantages:

- The simulation does conform closely to the actual implementation. Thus, it provides a relatively accurate model from the standpoint of logic and timing.
- When used as a tool in automatic testing and timing isolation, the diagnosis can be directly traced to the failing logic gates.
- Single fault modelings and test development algorithms using this technique have been well-developed over the years and are relatively simple to implement.
- A great deal of experience on this approach has been accumulated by test designers together with much literature and software support.

TABLE 9. SYSTEM CAPABILITY (GATE LEVEL)

System Name	System Maturity	IC Modeling Capability	Circuit Complexity	Circuit Modeling Convenience	Good Circuit Simulation Capability	Fault Simulation Capability	Fault Dictionary Data	Probe Data	ATG Capability
D4-LASAR	Very Mature	5	5	5	6	10	N Detects	Yes	8
FAIRTEST	Limited Use	3	5	4	6	7	1st Detects	Ø	5
CAFIG	Limited Use	2	1	3	4	3	As part of ATLAS test program	No	Ø
FANSIM	Mature	5	9	6	6	6	1st detect (sophisticated x-handling)	No	4

Ø = Capability not available

10 = Excellent

TABLE 10. GENERAL CONSIDERATIONS (GATE LEVEL)

System Name	Primary System Intent	Implementation Language	Host Computer	Interactive vs Batch	Present User Base	System Availability	Tester Outputs
D4-LASAR	Automatic Pattern Generation Fault Scoring (MFG, Field Service)	67% FORTRAN 33% Assembly Language and Microcode	Digitest SMC 3100 and Univac 1108 Mainframe	Interactive and Batch	1,000 Users including D2, D3 Versions	Commercial Time-Sharing or Purchase Object	Teradyne, AAI, RCA, VAST, HP, GenRad, TI, CAI, etc.
FAIRTEST	Test Program Generation for Fairchild	Assembly Language	IBM 360/370 Mainframe	Batch	5 Users	Commercial Time-Sharing or Purchase Object	Fairchild, Sentry, HP 9580
CAFIG	Test Program Generation	Assembly Language and FORTRAN	Bendix BDX 6200	?	Bendix Internal	Purchase	General Dynamics 12A6861, Bendix, 13A9070
FANSIM	IC Design and Test Program Generation	FORTRAN IV and Assembly Language	IBM 370 Mainframe	?	GTE Internal Use Only	Special Arrangement	ADAR IC Tester, TI9900, Systron-Donner 3700

TABLE 11. IC MODELING (GATE LEVEL)

System Name	Number of Primitives	Number of Unique IC Models in Library	Can User Model Primitives, ICs	Primitive Modeling Technique	IC Modeling Technique	Special LSI Capabilities
D4-LASAR	1 primitive (NAND Gate)	400	No/Yes	NAND gate only	Circuit Language, ICs and primitives	None
FAIRTEST	8 gates, also 3 complex functionals for good-circuit simulation	150	No/Yes	Boolean functionals	Circuit Language, use IBM Macro-assembler	None
CAFIG	5 Gates	150	No/Yes	Assembly Code	Circuit Language, ICs and primitives done only by Bendix	None
FANSIM	30 simple functionals plus RAM element	400	Yes/Yes	Boolean equation with 7 inputs 1 output	Circuit Language, primitives only	None

TABLE 12. CIRCUIT MODELING (GATE LEVEL)

System Name	Circuit Language	Circuit Capacity	IC Library Support	Automated Handling of Wired-ORS? ANDS	Automated Handling of Buses
DS-LASAR	Fair	Up to 15,000 gates on UUC TS 5,000, NANDS only	Very good, user has choice of libraries	No	No
FAIRTEST	Fair, IBM standard macro-assembly must be used first	Up to 15,000 gates, 32,000 nodes (recently expanded from 4,000 nodes)	Good	No	No
CAFIG	Fair	Number of gates + external ≤ 1,000	Fair, separate library IBM 379	No	No
FANSIM	Good	Up to 100,000 gates ≤ 80,000 primitives	Good	Yes	No

TABLE 13. GOOD-CIRCUIT SIMULATION (GATE LEVEL)

System Name	Test Programming Language	Logic States	Logic Timing Model	Timing/Race Model	Display of IC Data
D4-LASAR (P-400)	Fair, absolute binary	0, 1, X, $\overline{X}$	Zero/unit delay	Worst case ± % delay propagation pessimistic	Poor, absolute binary by user node
FAIRTEST	Very good incremental language incl. conditionals and subroutine calls	0, 1, X	Nominal delay for transition high, transition low	Oscillation detect only	Good, batch oriented
CAFIG	?	0, 1, X	Unit delay	Oscillation detection only	Poor, display of only one primitive at a time
FANSIM	Good, incremental waveform specification	0, 1, X	Nominal delays, transitions high and low	Special primitive will warn user of small pulses	Good, batch oriented, select desired nodes

TABLE 14. FAULT SIMULATION (GATE LEVEL)

System Name	Fault Analysis Technique	Fault Types	Fault Collapsing	Interactive Select/ Display	Fault Dictionary	Fault Resolution
D4-LASAR	Analytic backtrace fault analysis	Output sa∅, sal input sal input sa ∅ implicity internal IC faults	Partial	Yes/Yes	1st detect and N detects	Yes, extensive
FAIRTEST	255 faults in parallel; also serial simulator	Output sa∅, sal input sa∅, sal adj. pins short	Yes	Yes/No	1st detect	No
CAFIG	10 faults	Output sa∅, sal input sal	Yes	None	As explicit part of the ATLAS test program	No
FANSIM	Concurrent fault simulation	Output, sa∅, sal input sa∅, sal	No	Yes, No	1st detects sophisticated X-handling	No

However, there are also some serious disadvantages, especially with the advent of high density integrated circuit chips.

- A gate equivalent description of the device is not always found in the data sheet. Alternative descriptions derived from the specifications may be incomplete and are difficult to use for detailed analysis.
- Gate delays are not normally given in the description. In order to obtain accurate timing analysis, gate delays will have to be assigned correctly, based on additional information.
- Model preparation is a difficult and laborious process. Information on signal and gates internal to the device model has to be stored. Internal signal transitions have to be handled. All of these consume both time and memory.
- Depending on the circuit complexity, this approach may require a large volume of details to be simulated.

Functional Simulation

As the circuit size increases with the development of LSI and VLSI chips, the amount of simulation to be performed at the gate level tends to become overwhelming. In many cases, the gate level simulation often offers too much resolution and provides too much irrelevant data. The gate level approach becomes computationally inefficient and costly. A single chip may now exceed the entire capacity of some ATG systems if gate-level modeling is used. A transition to higher levels of simulation becomes necessary to reduce the number of details that need to be analyzed, to

simplify the model specifications, and to provide the needed flexibility to cope with technological changes. The functional simulation has become the more popular approach.

The functional simulation may be regarded as a pseudo gate-level approach in which the complexity of the basic building blocks has been increased from simple gates to complex functions. The complexity of these functional blocks often depends on the partitioning of the circuit functions because of packaging constraints. They may vary from simple functionals such as gates and flip-flops, to medium complex functionals such as RAMs, ROMs, shift registers, etc., to complex functionals such as ALUs, bit-sliced microprocessors, PIAs, etc. Thus typically SSI, MSI, and even LSI circuit packages may become the functional block. The more complex the basic functional block is, the greater the gain in computational efficiency. As the building block components achieve greater functional complexity, the need for simulating their internal workings diminishes. This in turn enables the functional simulation approach to become more desirable.

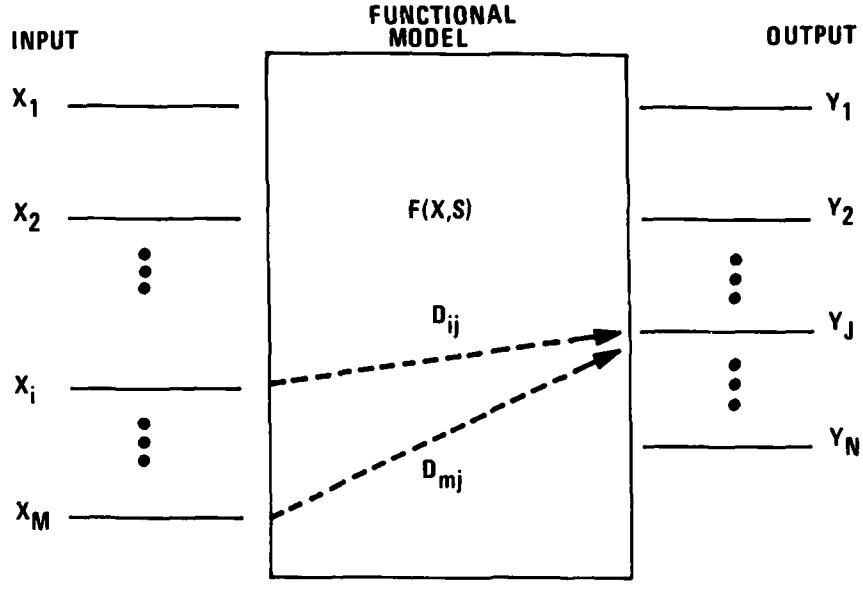
Simulation algorithms are used for these complex gates to produce the appropriate response on the gate outputs for each of the different input stimuli. There are several approaches in preparing these functional models. A functional model has inputs, internal state variables, and outputs. The response is then determined by the modeling algorithm.

One approach involves using a subroutine to solve logic equations written for the specific functional block. The subroutine will, in general, describe a logical function of combinational or sequential nature. Given the values for an input stimuli vector and the current internal states, the output response can be computed. This is shown in Figure 3.

The timing relationships, such as the propagation delays (D_{ij}), which depend on the way the signals propagate from inputs to outputs in the model, have to be considered and taken care of. With this approach, the modeling subroutine can be written with the level of accuracy given in the data sheet with regard to logic and timing relations. In addition, there are no internal signals or gates in the model which have to be handled other than the state variables, thereby saving memory space and processing timing. In general, these modeling programs are written in a type of high-level language to directly emulate the device behavior.

At the primitive gate level, only simple single-level Boolean functions are involved. However, at the complex gate level, the modeling algorithms must be able to deal with multiple levels of logic and may also be required to retain and use memory state values.

Another approach is to transpose the information in the data sheet concerning the logic and timing relations directly into a function table and then perform a table look-up operation to perform the simulation. In this way, the input values are directly or indirectly used to "address" the proper output response. In general, it is quite easy to write a function table description because most logical functions for a digital device are often provided in the form of a function table.



INPUT VECTOR $X = (X_1, X_2, \ldots X_M)$

OUTPUT VECTOR $Y = (Y_1, Y_2, \ldots Y_N)$

INTERNAL STATES $S = (S_1, S_2, \ldots S_L)$

Figure 3. Functional Simulation Representation

In modeling a device with the function table, many symbols must be allowed for. These symbols correspond to the more important symbols found in data sheets. The symbols are used to define input states or input state transitions and to trigger new events for the outputs. They include:

L	Low
H	High
Z	High Impedence—Tri-State
UN	Unknown
X	Irrelevant
PP	Positive Pulse
NP	Negative Pulse
↑	Positive Going Edge
↓	Negative Going Edge
NO...	No Change . . .
TO	Toggle
UNL	Transition Unknown to Low
UNH	Transition Unknown to High

The functional simulation approach is being used currently by many test development systems. This reflects the cost and difficulties associated with the gate-level approach in handling high density integrated circuits. In fact, the FAIRTEST and FANSIM systems described previously in the gate-level simulation section can actually be considered to have a partial functional simulation capability with the use of several complex gates as primitives.

From the literature, 25 test development and design aid systems have been identified to use the functional simulation approach. They include the following:

AAI Simulator	(AAI Corporation)
AIDS	(Applied Technology Inc.)
CAI Simulator	(Computer Automation)
CAPS II	(Gen Rad)
DFA	(Hughes Aircraft Company)
DIGISAT	(Hughes Aircraft Company)
FAST	(Honeywell Information Systems)
FLASH	(Micro Systems Division)
F/LOGIC	(Bell Northern Research)
GRASS	(Raytheon Company)
INDICATES	(Sperry Univac)
LAMP	(Bell Telephone Laboratories)
LOGCAP II	(National CSS)
LOGICV	(Westinghouse)
LOGOS III	(Grumman Aerospace Corporation)
LSP	(Naval Surface Weapons Center)
LSTV	(NCA Corporation)
MICROSIM	(Instrumentation Engineering)
NEWSIM	(Texas Instruments)
SALOGS	(Sandia Laboratories)
SIMSTRAN	(Rockwell International)
TASC	(Pacific Applied Systems)
TEGAS II	(Comprehensive Computing)
TESTAID III	(Hewlett Packard)
TESTGEN	(Hughes Ground Systems)

The capabilities and characteristics are summarized in Tables 15 through 20.

The content of these tables is extracted from "Selection Guide for Digital Test Program Generation Systems" by Henckels, Haas and Brown, Inc.

Compared to gate-level simulation, it usually takes a longer time to develop a system using the functional simulation approach, because it requires more advanced programming techniques, more sophisticated considerations, and more development. In order to allow complex LSI circuits to be modeled as functional blocks, some form of special high level functional modeling language or circuit description language is required to achieve modeling flexibility. The circuit is first described in this high level language; the language is then compiled for, or interpreted by, the simulator. In any event, it will generally require a highly skilled user to accurately transpose the logic diagram of an IC into a flow chart and then convert it into software program.

There are other additional drawbacks to the functional approach. The functional simulation is only an approximation to the actual overall design. There are fewer timing and implementation details in the functional models. Timing paths through the circuits are generally less accurately modeled than those given by the primitive gate level. In addition, the fault modeling becomes more complex. Because most internal gate faults are not modeled, the internal stuck fault coverage of the test generation algorithm may not be adequate. Diagnosis data generated from such analysis may not cover many internal package failures. Diagnosis of the failing primitive gate may no longer be possible.

TABLE 15. SYSTEM CAPABILITY (FUNCTIONAL)

System Name	System Maturity	IC Modeling Capability	Circuit Complexity	Circuit Modeling Convenience	Good Circuit Simulation Capability	Fault Simulation Capability	Fault Dictionary Data	Probe Data	ATG Capability
AIDS	New	6	9	6	10	0 not yet available	0	0	0
CAI SIMULATOR	Very Mature	5	6	8	5	9	1st detect	CAI Testers	0 not yet available
CAPS VIII	Very Mature	7	5	10	6	9	1st detect	GenRad Testers	2
FLASH	Mature	2	4	4	4	4	1st detect	Yes	0
LOGCAP	Very Mature	5	8	3	8	8	0	TI Tester	2
LOGOS III	Mature	5	6	5	5	8	N detects	0	8
MICROSIM	Mature	7	3	9	7	8	N detects	IE Testers	0
TASC	Very Mature	4	3	3	5	4	0	TI Tester	0

0 - Unavailable

TABLE 15. SYSTEM CAPABILITY (FUNCTIONAL) (continued)

System Name	System Maturity	IC Modeling Capability	Circuit Complexity	Circuit Modeling Convenience	Good Circuit Simulation Capability	Fault Simulation Capability	Fault Dictionary Data	Probe Data	ATG Capability
TEGAS III	Mature	6	9	3	8	9	N detects	0	4
TESTAID III	Very Mature	5	6	8	6	9	1st detect	HP Tester	4
AAI SIMULATOR	New	3	3	3	4	4	D-LASAR fault dictionary	No	0
DFA	Limited Use	4	3	4	5	4	1st detect	Yes	0
DIGISAT	Limited Use	4	5	6	5	4	N-detects (optional x-detect)	Yes	0
FAST	Mature	6	9	4	5	5	1st detect	No	6
F/LOGIC	Limited Use	4	9	5	10	7	1st detect (visual only)	No	2

0 = Unavailable

TABLE 15. SYSTEM CAPABILITY (FUNCTIONAL) (concluded)

System Name	System Maturity	IC Modeling Capability	Circuit Complexity	Circuit Modeling Convenience	Good Circuit Simulation Capability	Fault Simulation Capability	Fault Dictionary Data	Probe Data	ATG Capability
GRASS	Mature	7	8	9	9	5	No	No	2
INDICATES	Mature	4	5	7	5	7	N detects	Yes	7
LAMP	Very Mature	8	9	8	9	7	N detects	No	3
LOGIC V	Mature	5	4	6	6	6	N detects	Yes	New
LSP	Limited Use	4	6	3	6	6	None	No	0
LSTV	New	New	9	New	New	New	N. A.	No	0
NEWSIM	Mature	7	9	9	7	7	None	Yes	0
SALOGS	Limited Use	5	6	3	8	5	None	No	0
SIMSTRAN	Limited Use	1	8	2	7	6	None	No	0
TESTGEN	Limited Use	4	4	3	3	None, part of ATG	N. A.	No	5

0 = Unavailable

TABLE 16. GENERAL CONSIDERATIONS (FUNCTIONAL)

System Name	Primary System Intent	Implementation Language	Host Computer	Interactive vs Batch	Present User Base	System Availability	Tester Outputs
AIDS	Computer aided design	ALGOL-W	IBM 370 Mainframe	Interactive and batch	Internal use 1 outside user	Special arrangement	AAI AN/USM 449 (v) 5565 D-LASAR compatible
CAI SIMULATOR	Test program generation, for manufacturing on CAI testers	Assembly language	CAI/LSI-2 16 bits/word mini-computer	Interactive	60 users	Purchase with CAI test system, obtain simulator free of charge	CAI testers, test program outputs to GenRad, Teradyne
CAPS VIII	Test program generation, for manufacturing on GenRad testers	Assembly language	DEC PDP/8 12 bits/word mini-computer	Interactive and batch	500 users	Purchase object with GenRad system	GenRad testers
FLASH	Test program generation	FORTRAN	IBM 360/370 Mainframe also PDP 11/35 GA SPC 16	Batch	20 users	Purchase source and object	GenRad, MICRO CAI, Membrain Teradyne, etc.
LOGCAP	Computer aided design	DESTRAN (FORTRAN)	IBM 370 Univac 1108 Mainframes	Interactive and batch	50 users	Commercial time-sharing or purchase object	Fairchild Sentry, TI ATS 960, CAI, Systron Donner, Teradyne, etc.

TABLE 16. GENERAL CONSIDERATIONS (FUNCTIONAL) (continued)

System Name	Primary System Intent	Implementation Language	Host Computer	Interactive vs Batch	Present User Base	System Availability	Tester Outputs
LOGOS III	Automatic pattern generation fault scoring (MFG, Field Service)	FORTRAN (some Micro code on 8/32)	IBM 370 Mainframes and Interdata 7/32 and 8/32	Interactive and batch	3 users	Lease or purchase (incl. source)	HP 9500, VAST, CAI Grumman
MICROSIM	Test program generation, manufacturing on IE testers	Assembly language	Interdata 7/32 or 8/32 32-bit minicomputer	Interactive and batch	7 including internal	Purchase object with IE system	IE 390 tester, AAI tester, D-I ASAR compatible
TASC	Test program generation	Assembly language	TI 960 minicomputer	Interactive and batch	12 users plus heavy internal use	Purchase object	TI ATS 960, HP, Bendix, CAI, GenRad Teradyne, etc.
TEGAS III	Design verification and test program generation	ANSI FORTRAN	CDC 6600, IBM 370, Honeywell 6000 Univac 1108	Interactive and batch	13 users plus time-sharing access	Commercial time-sharing or purchase object	Fairchild Sentry Membrain
TESTAID III	Automatic test program generation for HP testers	Assembly code plus Micro code	HP 21 series minicomputer	Interactive and batch	100 users	Purchase object with HP System	HP-DTS 70 HP 9500s
AAI SIMULATOR	Test program generation (Field Service)	ANSI FORTRAN	Interdata 7/32 32-bit minicomputer, also PDP11	?	AAI internal use only	Special arrangement	AAI AN/USM 449 (v) 5565 D-I ASAR compatible output

TABLE 16. GENERAL CONSIDERATIONS (FUNCTIONAL) (continued)

System Name	Primary System Intent	Implementation Language	Host Computer	Interactive vs Batch	Present User Base	System Availability	Tester Outputs
DFA	Test program generation	80% FORTRAN 20% Assembly language	GA SPC 16 mini-computer	Batch	Hughes internal use only	Special arrangement	GenRad 1792, HP DTS 70, Hughes 1024
DIGISAT	Test program generation	75% FORTRAN 20% Assembly language	PDP-10 Mainframe	Interactive and batch	Hughes internal use only	Special arrangement	HP DTS 70, GenRad 1790, Tektronix 3260
FAST	Test program generation (Field Service)	80% Assembly language 15% COBOL 5% FORTRAN	Honeywell 6000/66 Mainframe	?	Honeywell internal use only	Special arrangement for time-shared access	In-house test systems only
F/LOGIC	Design verification	FORTRAN, some assembly language	IBM 370 Mainframe	Interactive and batch	Bell Northern internal use only	Special arrangement	Fairchild Sentry VII, CAI testers
GRASS	Test program generation	FORTRAN, some assembly language	Cyber 174	Batch	Raytheon internal use only	Special arrangement	HP 9500, Teradyne I.115 Fairchild Sentry VII, Several in-house
INDICATES	Test program generation, testability analysis	60% Assembly language 40% FORTRAN	UNIVAC 1100 series Mainframe	Interactive and batch	Sperry Univac internal time-share use only	Special arrangement	GenRad testers, Teradyne testers

TABLE 16. GENERAL CONSIDERATIONS (FUNCTIONAL) (continued)

System Name	Primary System Intent	Implementation Language	Host Computer	Interactive vs Batch	Present User Base	System Availability	Tester Outputs
LAMP	Fault coverage analysis	Assembly language	IBM 370 Mainframe	Interactive and batch	Bell internal time-share use, 100 users at a time	Special arrangement	Teradyne, Tektronix, GenRad, Bell Labs CPTEST
LOGIC V	Test program generation	FORTRAN IV, 1% Assembly language	UNIVAC 1100 Honeywell 6000 Mainframe	?	Westinghouse internal use only	Special arrangement	HP 9500, VAST, Grumman CAT, Grumman DTS
LSP	Fault coverage analysis	Extended FORTRAN	CDC 6600	?	Internal use only	Available to any gov't sponsored contractor	None
LSTV							
NEWSIM	Test program generation	60% FORTRAN VIII 40% Assembly language	IBM 370 Mainframe	Batch	T.I. internal time-share use only	Special arrangement	TI ATS 960

TABLE 16. GENERAL CONSIDERATIONS (FUNCTIONAL) (concluded)

System Name	Primary System Intent	Implementation Language	Host Computer	Interactive vs Batch	Present User Base	System Availability	Tester Outputs
SALOGS	Design aid for LSI	FORTRAN, some assembly	CDC 6600, PDP-10 Mainframe	?	Sandia Lab users	Available through NSWL	Fairchild Sentry 6000
SIMSTRAN	Design aid for LSI	Assembly language	IBM 370 Mainframe	?	Rockwell internal time-share use only	Special arrangement	Tektronix, Fairchild Sentry's, Rockwell in-house
TESTGEN	Automatic test generation only	PL/1	IBM 370	Batch	Hughes internal use only	Special arrangement	Hughes tester

TABLE 17. IC MODELING (FUNCTIONAL)

System Name	Number of Primitives	Number of Unique IC Models in Library	Can User Model Primitives, ICs	Primitive Modeling Technique	IC Modeling Technique	Special LSI Capabilities
AIDS	20 Complex functionals	250	No/Yes	ALGOL Code	Circuit language, ICs and primitives	ALGOL allows easy modeling of primitives
CAI SIMULATOR	ICs are modeled as truth tables	300	Yes/Yes	Truth table design	Truth table design, macro modeling available	Register transfer language (ICDL)
CAPS VIII	40 Complex functionals	500	No/Yes	Assembly code	Circuit language, ICs and primitives	Simulation command language (SCL)
FLASH	ICs are modeled as FORTRAN routines	200	Yes/Yes	FORTRAN routines	FORTRAN routines, need skills of a system programmer	None
LOGCAP	24 complex functionals	75	No/Yes	Destran (FORTRAN) language	Circuit language, ICs and primitives	None
LOGOS	20 simple functionals	100	No/Yes	FORTRAN code and assembly	Circuit language, primitives only	Special ROM and RAM modeling capabilities
MICROSIM	35 complex functionals	300	No/Yes	Assembly code	Circuit language, ICs and primitives	Microprocessor Control Module specification language (MCL)

TABLE 17. IC MODELING (FUNCTIONAL) (continued)

System Name	Number of Primitives	Number of Unique IC Models in Library	Can User Model Primitives, ICs	Primitive Modeling Technique	IC Modeling Technique	Special LSI Capabilities
TASC	100 medium functionals	None, library applications notes only	No/Yes	Assembly code	Circuit language, primitives only	None
TEGAS III	50 complex functionals	100	Yes/Yes	FORTRAN subroutines	Circuit language, primitives only	None
TESTAID III	15 medium functionals	300	No/Yes	Assembly code, single output primitives only	Circuit language, primitives only	None
AAI SIMULATOR	20 simple functionals	200	Yes/Yes	FORTRAN, also Boolean Equations	FORTRAN, also macro modeling	None
DFA	20 simple functionals	100	No/Yes	Assembly code	Circuit language, primitives only	None
DIGISAT	12 simple functionals	250	No/Yes	Assembly code	Logic equation language	None
FAST	ICs are modeled as truth tables	300	Yes/Yes	Truth tables, generated via standard macros	Special purpose high level language	None
F/LOGIC	25 complex functionals	100 Macro blocks used in LSI design	No/Yes	FORTRAN code	Circuit language ICs and primitives	None

TABLE 17. IC MODELING (FUNCTIONAL) (continued)

System Name	Number of Primitives	Number of Unique IC Models in Library	Can User Model Primitives ICs	Primitive Modeling Technique	IC Modeling Technique	Special LSI Capabilities
GRASS	40 complex functionals	800	No/Yes	FORTRAN code	Circuit language ICs and primitives	None
INDICATES	13 simple functionals	250	No/Yes	Assembly code	High level language, ICs and primitives	None
LAMP	10 simple functionals	300	Yes/Yes	Assembly code, also functional Description Language (FDL)	Circuit language, ICs and primitives	Functional Description Language (FDL) for register level modeling
LOGIC V	51 complex functionals	200	No/Yes	FORTRAN code	FORTRAN subroutines, also circuit language	None
LSP	25 complex functionals	No IC library	No/Yes	FORTRAN code	Circuit language, primitives only	State transition diagram specification for 8080A1
LSTV	30 complex functionals	None yet	Yes/Yes	FORTRAN code, Boolean equations	Circuit language, ICS and primitives	PLA primitive, also can accept binary IC models for proprirtary ICs

TABLE 17. IC MODELING (FUNCTIONAL) (concluded)

System Name	Number of Primitives	Number of Unique IC Models in Library	Can User Model Primitives, ICs	Primitive Modeling Technique	IC Modeling Technique	Special LSI Capabilities
NEASIM	400, most of which are ICs	500	No/Yes	FORTRAN sub-subroutines, also assembly code	Circuit language, ICs and primitives	None
SALOGS	50 medium functionals	100 Macro blocks (for LSI design)	No/Yes	FORTRAN code	Circuit language, ICs and primitives	None
SIMSTRAN	100 medium functionals (50 only for FAULTSIM)	None	No/No	Assembly code	LSI design tool, no IC concept exists	None
TESTGEN	30 simple functionals	50	No/Yes	PL/1 code	Circuit language, primitives only	None

TABLE 18. CIRCUIT MODELING (FUNCTIONAL)

System Name	Circuit Language	Circuit Capacity	IC Library Support	Automated Handling of Wired-ORS7 ANDS	Automated Handling of Buses
AIDS	Fair (Excellent for design)	Up to 100,000 gates, 65,000 nodes	Fair	No	No
CAI SIMULATOR	Good	Up to 25,000 gates, 4,096 nodes	Good	Yes	Yes
CAPS VIII	Very good	Up to 15,000 gates. Limited to approx. 2,000 nodes by 32k words of memory	Very good, user selects IC libraries	Yes	Yes
FLASH	Fair	Up to 7,500 gates, 1,000 nodes	Fair	No	No
LOGCAP	Fair primitives only	Up to 75,000 gates, 32,000 nodes	Fair	No	No
LOGOS	Fair	Up to 25,000 gates, 16,000 nodes, 11,000 primitives	Good	No	If specified by user as bus pin
MICROSIM	Very good	Up to 75,000 gates, 16,000 nodes	Good	Yes	Yes
TASC	Fair, primitives only	Up to 5,000 gates, 1,024 nodes	No library	No	No

TABLE 18. CIRCUIT MODELING (FUNCTIONAL) (continued)

System Name	Circuit Language	Circuit Capacity	IC Library Support	Automated Handling of Wired-OR's/ANDS	Automated Handling of Buses
TEGAS III	Fair, primitives only	Up to 100,000 gates, 40,000 primitives	Fair	No	No
TESTAID	Very good	Up to 25,000 gates, 11,000 primitives	Very good, 4 libraries	Yes	No
AAI SIMULATOR	Fair, D-LASAR language	Up to 5,000 gates, 1,200 nodes	Fair	No	No
DFA	Good	Up to 5,000 gates, 2,000 nodes	Fair	No	No
DIGISAT	Good	Up to 15,000 gates	Good	Yes	No
FAST	Fair	Up to 100,000 gates, ≤ 40,000 primitives	Good	No	No
F/LOGIC	Fair (Good for chip design)	Up to 100,000 gates, 32,000 primitives	Good	No	Only by special modeling using "switch element"
GRASS	Good	Up to 75,000 gates	Very good, user has private library as well as main one.	No	Yes
INDICATES	Good	Up to 15,000 gates, 7,000 primitives	Good	No	Partial

TABLE 18. CIRCUIT MODELING (FUNCTIONAL) (concluded)

System Name	Circuit Language	Circuit Capacity	IC Library Support	Automated Handling of Wired-ORS? ANDS	Automated Handling of Buses
LAMP	Very good	Up to 100,000 gates, 6,000 nodes	Good	Yes	Yes
LOGIC V	Good, also partial analog modeling	Up to 7,500 gates	Fair	No	No
LSP	Fair, primitives only	Up to 25,000 gates, 8,000 nodes, 100 primitive types	None	Yes	No
LSTV	Good	Up to 100,000 gates	Not yet	No	No
NEWSIM	Very good	Up to 100,000 gates	Good	Yes	Yes
SALOGS	Fair, (Good for LSI design), primitives only	Up to 25,000 gates, 20,000 primitives only	Fair	No	No
SIMSTRAN	Fair (Good for LSI design), primitives only	Up to 75,000 gates	None	No	No
TESTGEN	Fair	Up to 7,500 gates, 4,000 primitives	Fair	No	No

TABLE 19. GOOD-CIRCUIT SIMULATION (FUNCTIONAL)

System Name	Test Programming Language	Logic States	Logic Timing Model	Timing/Race Model	Display of IC Data
AIDS	Very good, high level control language	12 states	Min/max rise/fall other program parameters	Full ambiguity region analysis	Very good
CAI SIMULATOR	Very good incremental language	0, 1, X	Zero/unit delay	Multiple transitions flagged	Very good, interactive display by IC and step
CAPS VIII	Very good incremental language. Macro capability	0, 1, X	Unit delay	Detects simultaneous transitions on FF, latches	Very good, interactive display by IC and step
FLASH	Good incremental language	0, 1	Unit delay	Skew analysis on edge connected	Good, batch oriented
LOGCAP	Good incremental waveform specification	0, 1, X, Z	Nominal delay for rise and fall times	Multiple transition analysis	Very good, display by node and time
LOGOS	Good, binary language with macro capability	0, 1, X	Unit delay	Near simultaneous transitions on sequential devices, generates message and unknowns (user specifiable)	Good, complete display
MICROSIM	Good incremental language	0, 1, Z	Nominal delay on outputs	Detects simultaneous transitions on FFs	Excellent, powerful interactive display

TABLE 19. GOOD-CIRCUIT SIMULATION (FUNCTIONAL) (continued)

[illegible]	Test Programming Language	Logic States	Logic Timing Model	Timing/Race Model	Display of IC Data
LASAR	?	0, 1, X, ↑, ↓	Zero/unit delay	Eichelberger race detection	?
[illegible]	Good incremental waveform specification	0, 1, X, ↑, ↓	Min/max, rise/fall propagation delays	Full ambiguity region analysis	?
[illegible]	Very good high level control language	0, 1, X	Zero/unit delay	Eichelberger race detection	Good, complete display
[illegible]	Basic absolute binary	0, 1, X	Zero/unit delay, also programmable delay	Detect simultaneous transitions on flip-flops	Good, including waveform display
[illegible]	?	0, 1, X, ↑, ↓	Zero/unit delay	Pulse data analysis	?
[illegible]	?	0, 1, X, ↑, ↓	Zero/unit delay	Pulse data analysis	?
[illegible]	Good incremental language	0, 1, X	Zero unit delay, also programmable delay element	Cycling detection and analysis. Monte Carlo Race analysis	Good, batch oriented, complete and partial displays
[illegible]	Very good incremental language incl. list processing	0, 1, X, Z, ↑, ↓	Min/max, rise/fall times, very flexible	Inherent in timing model	Very good, interactive, select partial or full display

TABLE 19. GOOD-CIRCUIT SIMULATION (FUNCTIONAL) (continued)

System Name	Test Programming Language	Logic States	Logic Timing Model	Timing/Race Model	Display of IC Data
GRASS	Good, waveform specification	0, 1, X, Z, ↑, ↓	Min/max rise/fall	Inherent in timing model	Good, batch oriented, select time range and desired nodes
INDICATES	?	0, 1, X, Z	Nominal delay of up to 64 units	Transient analysis, check for illegal sequencing	?
LAMP	Very good incremental language, incl. loops, calls	0, 1, X, ↑, ↓	Min/max rise and fall times	Full ambiguity region analysis	?
LOGIC V	Good, incremental waveform specification	0, 1, X	Unit delay, also can schedule events via cause/ cancel statement	Oscillation detection only	Good, timing diagram display
LSP	Good, incremental vector oriented language	0, 1, X	Min/max rise and fall times	Race analysis for NAND/NOR latches	Good, timing diagram display
LSTV	Good incremental language	16 states	Min/max rise and fall times	Inherent in timing model, multiple	Very good, display by IC and step
NEWSIM	Very good incremental language incl. clock generators	0, 1, X, Z	Nominal propagation	Eichelberger race detection	Good, batch oriented

TABLE 19. GOOD-CIRCUIT SIMULATION (FUNCTIONAL) (concluded)

System Name	Test Programming Language	Logic States	Logic Timing Model	Timing/Race Model	Display of IC Data
SALOGS	Good, flexible incremental language	8 states	Zero/nominal delay. Delay element with 19 units for rise and fall times	Simultaneous transitions on gates are flagged via messages	Good, batch oriented display by nodes
SIMSTRAN	Very good incremental language incl. list processing	0, 1, X, Z	Nominal rise and fall times	None	?
TESTGEN	Automatic vector generation only	0, 1, X	Zero/unit delay	None	Poor, dump of node states

TABLE 20. FAULT SIMULATION (FUNCTIONAL)

System Name	Fault Analysis Technique	Fault Types	Fault Collapsing	Interactive Select/Display	Fault Dictionary	Fault Resolution
AIDS	Deductive fault list approach in development	N. A.	N. A.	N. A.	N. A.	N. A.
CAI SIMULATOR	Simulated fault list	Output sa∅, sal Input sa∅, sal Limited internal IC faulting	Yes	Yes/Yes	1st detect (optional x detect)	No
CAPS VIII	6 faults in parallel, serial for large RAM	Output sa∅, sal Input sa∅, sal Adj. pins shorts, multiple IC faults	Yes	Extensive fault selection, no display	1st detect	Yes
FLASH	Serial fault simulation (no x-simulation)	Output sa∅, sal Input sal Adj. pins shorts VOC GND missing	No	Yes/Yes	1st detect	No
LOGCAP	Simulated fault list	Output sa∅, sal Input sa∅, sal Internal IC faults	Yes	Yes/No	No	No
LOGOS	LOGOS II: 32 faults in parallel LOGOS III: deductive fault list	Output sa∅, sal Input sa∅, sal shorts, dynamic faults	Yes	LOGOS II: No/No LOGOS III: Yes/Yes	Up to complete isolation	Yes
MICROSIM	Serial fault simulation	Output sa∅, sal Input sa∅, sal and pins disconnected	Yes	Yes/Yes Extensive, flexible	N-detects	No

TABLE 20. FAULT SIMULATION (FUNCTIONAL) (continued)

System Name	Fault Analysis Technique	Fault Types	Fault Collapsing	Interactive Select/Display	Fault Dictionary	Fault Resolution
TASC	Serial fault simulation	Output sa∅, sal Input sa∅, sal Internal IC faults	Partial	Yes/Yes	No	No
TEGAS III	(N-1) faults in parallel where N = # bits in word	Output sa∅, sal Input sa∅, sal internal faulting	Yes	No/No	N-detects	Yes
TESTAID III	8 faults in parallel & topologically disjoint faults	Output sa∅, sal Input sa∅, sal	Yes	Yes/Yes limited	1st detect (N-detects for x-detects)	Yes
AAI SIMULATOR	Boolean fault list	Output sa∅, sal Input sal Internal IC faults	No	No/No	D-I ASAR fault dictionary	Yes
DFA	Serial fault simulator	Output sa∅, sal Input sa∅, sal	Yes	No/No	1st detect	Yes
DIGISAT	17 faults in parallel	Output sa∅, sal Input sa∅, sal Internal IC faults	Yes	No/No	N-detects (optional x-detects)	Yes
FAST	35 faults in parallel (no x-simulation)	Output sa∅, sal Input sal Adj. pins short	Yes	Yes/No	1st detect	Yes
F/I OGIC	1023 faults in parallel (fewer for large boards)	Output sa∅, sal Input sa∅, sal Shorts by specific request	Yes	Yes/Yes flexible	1st detect (presently only in visual form)	No

TABLE 20. FAULT SIMULATION (FUNCTIONAL) (continued)

System Name	Fault Analysis Technique	Fault Types	Fault Collapsing	Interactive Select/Display	Fault Dictionary	Fault Resolution
GRASS	59 faults in parallel simulates 6 gates, primitives only	Output sa0, sa1 Input sa0, sa1 Internal IC faults (selectable)	Yes	No/No	No	No
INDICATES	35 faults in parallel	Output sa0, sa1 Input sa0, sa1 Internal IC faulting	Yes	No/No	N-detects	No
LAMP	Deductive fault list	Output sa0, sa1 Input sa1 Internal IC faulting	Yes	Yes/Yes (indirect)	N-detects	No
LOGIC	[illegible]	Output sa0, sa1 Input sa0, sa1	Yes	Yes/on edge only	N-detects	Yes
[illegible]	[illegible]	Output sa0, sa1 Input sa0, sa1	Yes	Yes/Yes	No	No
[illegible]	[illegible]	Output sa0, sa1 Input sa0, sa1 Internal IC faulting	Yes	Yes/No	Yes	No
[illegible]	[illegible]	Output sa0, sa1 Input sa0, sa1 Memory sa0, sa1 Internal IC faults	Yes	No/No	No	No

TABLE 20. FAULT SIMULATION (FUNCTIONAL) (concluded)

System Name	Fault Analysis Technique	Fault Types	Fault Collapsing	Interactive Select/Display	Fault Dictionary	Fault Resolution
SALOGS	29 faults in parallel	Output sa∅, sal Input sa∅, sal Internal faults (via fault table)	No	Yes/No	No	No
SIMSTRAN	Analytical backtrace fault analysis	Output sa∅, sal Input sa∅, sal	No	No/No	No	No
TESTGEN	Done as part of automatic vector generation	N. A.	N. A.	N. A.	N. A.	N. A.

However the advantages of the functional approach in many ways outweigh its disadvantages, particularly for high density complex circuits. The key is in computational efficiency. With today's IC technology, the size of a UUT modeled in terms of logic gates can become too large. Simulation of a complex functional block may be slower than that of a primitive gate, but it is still much faster than simulating the tens of thousands of primitive gates that might otherwise be required at the gate level approach. The functional modeling approach also requires less storage space and can handle test generation on functional design in a much more powerful way for high density chips such as ROM, RAM, etc.

The other advantages include the following:

- The functional information is much easier to obtain from the semiconductor vendors than that for gate-level models.
- Global circuit controls are easier to identify.
- There is a large reduction in the volume of details to be processed.
- Diagnosis to functional subunits is possible.
- Some analog circuits can also be simulated.

There can be several degrees or levels of functional simulation, ranging from the modeling of primitive IC gates as Boolean functions to the abstract description of the total circuit function. The selection of the level of simulation is then dependent on the trade-off of storage space and execution time vs accuracy and completeness.

With the development of LSI and VLSI technology, there is growing support for the higher level approach, also known as behavioral simulation.

Interface-Level Simulation

The interface-level simulation can be considered as a subset of the functional level simulation, whereby a table look-up technique is used to implement the simulation. This approach has been widely used in areas such as trainer simulation systems. By storing the appropriate output values or responses for each possible combination of inputs in a set of tables, this approach can provide a limited simulation capability on the behavior of the UUT. In general only static, typically table-driven, input/output relationships are assumed.

A table-driven circuit simulator operates upon the topology of the circuit. The circuit is stored as a set of tables indicating information such as fan-in list, fan-out list, signal or logic values, types of gates, etc. For a given set of input values, a set of output values can be determined, which in turn will become the inputs for the next table. In this manner, a simple dynamic simulation system can be implemented.

Simple fault detection and fault simulation can also be carried out by determining the outputs for a sequence of input signals. The output values can then in turn be used in conjunction with a fault signature table for fault detection or simulation. In general, some form of special purpose language or processing will be required to support these additional entries of input/output relationship tables into the simulation system.

The interface level of simulation cannot be used to provide details on implementation and timing analysis. However, it does offer the following advantages:

- It does not require skilled users.
- The computer resource required is relatively small.
- Fast simulation time is possible.

Mixed-Level Simulation

With the advent of high density chips such as VLSI and VHSI circuits and also hybrid (analog/digital) systems, a new approach called mixed-level (or multi-level) simulation has been proposed, whereby simulations can be performed at any one of the following three levels: analog, gate and functional, or a combination of them.

For the proposed system, a circuit may be regarded as a set of interconnecting blocks. For each block, simulation can be carried out at various levels of detail, from the abstract algorithmic functional level to the detailed electrical analog modeling. Simultaneous simulation of a given block at several levels can also be performed.

The main advantage of having the multi-level capability is that it enables simulation of a system to be performed in a top-down hierarchical manner. The entire system can be simulated even though parts of the design are still in different stages of development. Furthermore, this technique may reduce the analysis of hybrid (mixed analog and digital) systems.

Simulation systems using this approach are currently under development by several companies, most notably by Texas Instruments. An earlier system that had been developed with a somewhat similar intention is the LOGIC IV simulator system by Westinghouse. The system was laid out as a hierarchical system design tool, whereby the design engineer first simulates the system at the highest level as a set of subroutines and then later on at lower levels in which the various subroutines are replaced by simulated subassemblies, such as functional digital circuits and even some analog circuits. The simulator allows for partial analog simulation because the circuit is modeled by using a FORTRAN subroutine, and the signal values can be stored as a 36-bit word rather than as a 2-bit number.

The mixed-level simulator under development at Texas Instruments, called INTSIM, can provide several simulation levels:

- Behavioral (simulation values 0, 1)
- Logic (multiple simulation values)
- Electrical (a modified version of SPICE 2)

A hierarchical Hardware Description Language (HDL) is used to drive the simulator and to support descriptions of the design performance requirements and design description data at every level, from behavioral and block functional to logic and electrical. The HDL source statements are then compiled into a hierarchical design data base. The Circuit Selection Language (CSL) is used to reverse compile and generate the HDL source from the design data base because it may contain the design at several levels of detail. Using CSL, the blocks to be simulated are selected from the design data base and a translation into simulation data structures is

performed. The Simulation Control Language (SIMCL) is used to specify the external environment, input stimuli, run options, the timing mode, and the signals to be displayed. The INTSIM can then be executed and the results evaluated. Changes can be made to the design (using HDL), circuit blocks (using CSL), and external environment (using SIMCL) and the simulation repeated.

INTSIM is intended for use in the following:

- Specification development
- Algorithm checkout
- Verification of specific implementation
- Simulation of test patterns
- Race/hazard timing analysis

However, there are some difficult issues that will have to be resolved before the mixed-level simulation technique can be fully operational. At present, a variety of programming languages are needed to use the system. Hardware Description Language (HDL) is used to describe the design, Circuit Selection Language (CSL) is used to select the parts for simulation, and Simulation Control Language (SIMCL) is used to specify the external stimulus for simulation. For the system to be user-oriented, it will require a great deal of developmental effort to arrive at a syntactically consistent form common to all three languages. A more detailed description of INTSIM can be found in the references.

Additional Information From Literature

An extensive search was made of the literature for simulation techniques that may be applicable to UUT simulation for V&V. The results are summarized in Table 21.

The purpose of this table is to help the reader find information on a specific simulation method. Entries in the table can help one determine if the referenced article contains simulation information of interest. The table is organized as follows:

- The principal author's name
- Title
- Method of simulation

There are five basic types of simulations referred to in the articles. They were assigned numbers 1 through 5 corresponding to the following list:

1. Analog circuit level simulation
2. Digital gate level simulation
3. Functional simulation
4. Interface level simulation
5. Multilevel simulation

Many articles contained information on more than one type of simulation and therefore have several numbers given. The numbers are not listed in any significant order.

TABLE 21. REFERENCE LITERATURE SUMMARY

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
2	Akers	Partitioning for Testability	1, 2	ATS	A test flow model for partitioning problems
5	Barbacci	Instruction Set Processor Specifications for Simulation, Evaluation, and Synthesis	3		ISP: an abstract design language, independent of structure or implementation of system; ISP deals with architecture and behavior of the system
6	Bastian	Nonlinear Device Modeling	1	ATS	Nonlinear CAD-type models are adequate; functional models due to lack of range are inadequate
13	Breuer	Test/80: A Proposal for an Advanced Automatic Test Generation System	3	DATPG	A description of Test/80 for digital ATG
20	Carter	Symbolic Simulation for Correct Machine Design	2, 3		Requires formal description of device; uses symbols rather than actual values for program variables or machine components
23	Chang	Automatic Test Program Generation	1	ATPG	NOPAL processor automatically produces test programs from specifications in NOPAL
24	Chien	On the Use of Deductive Models for Automatic Generation of Test Program for Analog Circuits	1, 2, 3, 5	ATPG	Deductive models of partitioned subcircuit; contains following information about circuits: • Physical characteristics • Functional • Diagnosis

TABLE 21. REFERENCE LITERATURE SUMMARY (continued)

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
25	Chien	Strategy for Automatic Testing by Circuit Understanding		ATPG	Strategy for functionally isolating a faulty circuit to one particular subcircuit.
32	Cortner	A User's View of the Next Step in Test Generation Software	2, 3	DATPG	RTL for modeling LSI and VLSI
37	Dejka	A Review of Measures of Testability for Analog Systems	1	AATPG	Linear and nonlinear circuits analysis programs
47	El-Zig	Testing of MOS Combination Networks: A Procedure for Efficient Fault Simulation and Test Generation	2		Cell matrix model; used to generate test vectors for fault detection and isolation of complex MOS cells
58	Giambiasi	SILOG: A Practical Tool for Large Digital Network Simulations	2, 3		Temporal and logic simulator; a deductive fault simulator based on dynamic event propagation 5000 gates can be simulated at low cost
60	Goodenough	Simulation Higher Order Language Requirements Study	3, 4		Defines simulator HOL requirements; develops a general approach to determine HOL requirements in a specific area (generic flight trainer simulator); three sources of language requirements: • Programming environment • Functional requirement • Language design principles All languages failed to satisfy some of the simulator language requirements.

TABLE 21. REFERENCE LITERATURE SUMMARY (continued)

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
63	Gracia	Multiplex Simulator Design Study	2		MUXSIM CAD for simulation of multiplex systems
75	Henckels	Evaluation Criteria for Test Program Generation Systems	2, 3	DATPG	Criteria digital test program generation systems
76	Henckels	Selection Guide for Digital Test Program Generation Systems	2, 3	DATPG	Techniques of digital ATPG; evaluation of 29 DATPG systems and their simulation methods
77	Henckels	Selection Guide for Digital Test Program Generation Systems (Appendices)	2, 3	ATPG	Summarizes 29 digital simulation systems and ATPG systems
80	Hill	SABLE: A Tool for Generating Structured Multi-Level Simulations	2, 3		SABLE (structure and behavior linking environment): a system to support multi-level simulation; description written in ADLIB (a design language for indicating behavior)
87	Johnson	Mixed-Level Simulation From A Hierarchical CHDL	1, 2, 3, 5		Hardware description language drives mixed level simulator INTSIM; several simulation levels
90	Kaplan	Computer-Aided Design	1		Discusses CAD system capabilities and analysis methods; 27 CAD systems are examined and compared

TABLE 21. REFERENCE LITERATURE SUMMARY (continued)

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
96	Kjelkerud	Methods of Modeling Digital Devices for Logic Simulation	2, 3		MISSIM SW for digital simulation and test-generation models: • Gate equivalent • Subroutine • Function table • Macro
103	Liu	Fault Diagnosis of Large-Scale Analog Systems: A Tearing Method Approach	1	ATS	Time domain, state equation approach to decomposition of circuits into circuit modules; theoretical discussion on the conditions for tearing
108	Manigian	Interactive Fault Trace: A Simulator Aid for Test Program Development	2, 3, 5	ATS	HOL for circuit schematic; IC models at circuit level
114	Modi	Navy Program for Development of an Analog Test Program Generation System	1	AATPG	Methods: • Volterra-Laplace differential equations • Adjoining circuit • State variable model
123	Nagel	SPICE 2: A Computer Program to Simulate Semiconductor Circuits	1		Implementation of numerical methods are described and compared; theoretical and practical aspects of SPICE 1 and SPICE 2 are documented; program can determine: • Quiescient operating points • Time-domain response • Small signal frequency domain response Contains models for common circuit components; can simulate most electronic circuits

TABLE 21. REFERENCE LITERATURE SUMMARY (continued)

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
124	Navabi	Efficient Simulation of AHPL	3		Hardware description language, HPSIM2, software structure, and communication section
129	Nuspl	An Incremental Charge Method for the Analysis of Nonlinear Circuits	1		Incremental charge method for time-domain analysis of nonlinear circuits; includes circuits with diodes and transistors
134	Pearson	Analysis of Software Simulation in Computer-Based Electronic Equipment Maintenance Trainer	1, 2, 3		Information on suppliers of CAD systems
136	Plice	Digital Filters as Analog Circuit Models	1	AATPG	Recursive difference equations into matrix state equations for computational efficiency
137	Plice	I-B Automatic Test Program Generation Recommendations	1, 2, 3	ATPG	Analog models needed: • CAD for dc, ac, and transient models • Digital filters • Complimentary signal design
138	Plice	Overview of Current Automatic Analog Test Design	1	AATPG	Types: • dc simulation (Syscap II) • ac steady state simulation • Transient simulator
141	Purks	Experiences with ATE Providing Testability of Microprocessor Boards	3, 4	ATS	Functional modeling for complex LSI

TABLE 21. REFERENCE LITERATURE SUMMARY (continued)

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
142	Raymond	CAPS: An Enhanced Interactive Simulation	2	ATPG	Computer program designed to duplicate in software the response generated in hardware: • Scores % of faults detected • Doesn t require working board • Complete diagnostic data base
148		RTL CAD: Computer Aided Design at the Register Transfer Level	3		System of programs HICAD (Honeywell Integrated CAD); deals with subsystem RTL CAD (set of programs for formulation and verification of digital device structure and operation) various levels in RTL language; sim. by program that creates model from RT description
159	Schreiber	A State Space Approach to AATG	1	AATPG	CAP-ITRAC
160	Schreiber	Fault Dictionary Construction Using the Complementary Signal	1	AATPG	Time-domain simulation uses extant CAP; avoids use of transfer functions and state transition matrix
161	Schreiber	An Overview of Analog Fault Isolation Techniques	1	AATPG	No extant circuit analysis program for EMI; simulation of failed components
162	Schreiber	A Review of AATG	2	AATPG	ATE immittance measurements; augmenting software system computes component values
169	Sherwood	A Hybrid Scheduling Technique for Hierarchical Logic Simulators	2, 3		Mixed level simulation: • Register transfer • Gate level

TABLE 21. REFERENCE LITERATURE SUMMARY (continued)

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
171	Sick	The Impact of Complex Digital UUTs on the Use of Computer Programs to Automatically Generate Test Programs	2, 3	DATPG	Modeling methods: • Gate level for SSI and MSI • Functional truth table simulation of logic elements
175	Thomas	Gate-Level Modeling for Digital ATPG	2, 3	DATPG	Simulator requirement for digital ATG
176	Thompson	Digital Logic Simulation Aids Utilized at McDonnell Aircraft Company (MCAIR)	2, 3	ATS	MCAIR's digital logic simulation programs
177	Timoc	Fault Simulation: An Implementation into Hardware			Hardware fault simulator for large sequential networks provides a two-order of magnitude improvement in simulation time over software simulators
179	To	Automatic Test Systems (Circuit Testing)	1, 2, 3, 5	ATS	Circuit board level
182	Tucker	Computer-Aided Design Application to S-3A Analog Test Programs	1	ATS	Steps for device modeling: • Find equivalent circuit • Determine parameter values Transistor models: • Ebers-Moll • Gummel-Poon
183	vanCleemput	An Hierarchical Language for the Structural Description of Digital Systems	2, 3		SDL: a language for describing structural properties of digital systems at various levels

TABLE 21. REFERENCE LITERATURE SUMMARY (concluded)

#	AUTHOR	TITLE	SIMULATION TECHNIQUE	CONTEXT	COMMENTS
186	Vartanian	A Program for Fault Analysis of Analog Networks	1	ATS	CAD (ECAP) program, circuit schematics, and component values to model UUT
189	Wang	Digital System Modeling for ATG	2, 3, 4	DATPG	Digital functional level modeling
194	Whisnant	AFAL Simulation Facility/Capability Manual	1, 2, 3, 5		Documents total simulation capability in a manner which will serve: • AFAL directorate • Potential and actual users of AFAL simulation facilities • AFAL staff involved with support

- Context

  The context refers to what purpose the simulation serves. The simulation is often used for a specific purpose like automatic test program generation (ATPG). If no particular use is given for the simulation, the context column is left blank. This usually indicates that the entire article was dedicated to simulation.

- Comment

  The comment describes the method of simulation referred to in the article. The objective is to give a summary view of the simulation technique used.

APPLICABLE TECHNOLOGIES

The purpose of this section is to discuss technology constraints to be used for the implementation of UUT simulators for V&V. The current rapid advancement of digital technology dictates that specific periods be targeted for the simulator implementation. Otherwise, if a technology is chosen that is too near-in, not enough performance may be available; if a technology that is too far out is chosen, projected performance may be adequate, but too many additional risk factors would be brought into the development of the simulator.

There are basically two time frames of interest for a UUT simulator in support of the MATE program. A UUT simulator demonstration system to be available in the 1983 time frame must use technology that is available in the 1981 time frame. A generic UUT simulator system in the post-1985 time frame, may be recommended as a result of the demonstration system to develop.

To help discuss the technology constraints, the following set of categories will be used:

- Computer Hardware
- Interface Devices
- Communication Support
- System Architecture
- Software
- Data Bases

Computer Hardware

Processors—For the 1983 time frame, a wide range of processors ranging from bit slice Arithmetic Logic Units through 8- and 16-bit micros to 16- and 32-bit minis will be capable of being used. All these technologies are in place today to varying degrees of completeness. The 16-bit micros should be available for use by the 1981 time frame. Augmentation support chips (such as floating point processors) are also available to support building special architectures for increasing simulation speeds. A speed range of 0.5 to 4 million instructions per second should be supportable.

For the post-1985 time frame, a full range of 8-, 16-, and 32-bit conventional processors as well as special purpose processors such as floating point units and signal processing and array processing units can be assumed available. Speed ranges in the 1 to 10 million instructions per second range can be assumed.

Main Memory—For the 1983 time frame, 16K to 64K RAM chips with 100 to 300 nsec. aceess can be assumed. Although the 64K RAM chips are not yet available, they can be assumed ready for use in 1982. The main implication in the use of a UUT simulator is that large main memories can be assumed available to the extent they are needed.

For the post-1985 time frame, 64K to 256K RAM chips can be assumed available with further improved access speeds.

Mass Storage—In the 1983 time frame, floppy disks with capacity ranges of 100K bytes to 2M bytes can be assumed available to support storage of UUT simulator characteristics. Cartridge disks and hard disks with storage capacities in the range 10M to 500M bytes can also be assumed available if large data bases are needed. However, storage requirements beyond the 20M to 50M byte range may imply that the implemented system is no longer portable.

In the post-1985 time frame, further increases in capacities are expected. Furthermore, magnetic bubble mass storage devices in the 1M byte to 50M byte range can be assumed available and cost-effective if the access time of the rotating media devices becomes a constraining factor in the UUT simulator performance.

Peripherals—A UUT simulator is not expected to require unusual peripherals. Standard tapes may be desirable to support backup and to capture trace information. Conventional printers and CRTs are assumed adequate for operator and user interfaces.

Interfaces---The main UUT simulator interface devices of concern are analog to digital and digital to analog converters. If a simulator solution requires interception and generation of direct analog signals, the accuracy, the speed, and the drive/sink capabilities of the converters are important. In the near term, one can assume 12-bit A/D converters with speeds of 1 MHz to 5 MHz are cost-effective to use in a simulator. Converters with less accuracy can run at higher speeds. D/A converters can run at higher speeds than A/Ds, but the nature of their output is more critical to their performance. To make full use of high converter speeds, very high-speed simulation processing would be required. The specific characteristics of the converters are less of a concern because the processing power would probably become a UUT simulator system bottleneck much earlier than the speed at which the conversions are performed.

Communication—In the 1983 time frame, conventional communication chips supporting RS232, IEEE-488, and other special communication protocols can be assumed available to support interconnecting sub-systems of a UUT simulator if desired.

In the post-1985 time frame, higher bandwidth optical links should be available. High-speed direct links to large data bases (such as would contain device characteristics and UUT design information) can be assumed available.

System Architecture—In the 1983 time frame, the following ranges of system architectures can be assumed available for use in a UUT simulator:

- Single computer systems (simplest case)
- Multi-computers cooperating on simulation support
- Limited forms of distributed processing

More advanced architectures such as parallel processors, pipelined processors and multi-micro-implemented simulators should not be considered in this time frame because they would introduce additional risks in demonstrating and evaluating the UUT simulator concepts.

In the post-1985 time frame, all of the above advanced architectures can also be used. In addition, it can be assumed that VLSI implemented "Direct Execution Machines" specifically oriented to supporting UUT simulations can be built into system architectures. These components would both improve the response times of the simulators and allow more cost-effective use of the simulation system.

Software—Programs developed for a 1983 time frame system would have to rely on software technology already currently in place. Since the UUT simulator would more than likely be based on a minicomputer, the implication is that the software would have to be FORTRAN-based. Test software is assumed to be based on ATLAS. Not much change for conventional support software available today is expected during the next few years.

In the post-1985 time frame, the DoD high order language ADA should be sufficiently mature and adequately supported to assume it could be used in further UUT simulator software development.

Data Bases—In the post-1985 time frame, it can be assumed that large data bases supporting UUT component models, UUT design data, and test program information will likely be available and conveniently accessible. The UUT simulator should therefore be capable of being built and driven by information from these data bases. The short term implication is that the UUT simulator design should be modular and not tied in to a specific method of simulation.

SOLUTION CATEGORIES

Now that we have the UUT simulator requirements, the applicable simulation techniques, and the assumed technology constraints, we can proceed to postulate what the UUT simulator system will look like. Because there are many possible solutions, some method of structuring them must be used in order to talk about them intelligently. The strategy we used was to look at the solutions in terms of orthogonal attributes.

The first dimension of this attribute space was assumed to be the level of simulation used in the UUT simulator:

1. Analog Circuit Level
2. Digital Gate Level
3. Functional Level
4. Interface Level
5. Multi-Level

The first 4 levels represent decreasing levels of fidelity as one proceeds from the analog to the interface level. The last case represents a combination of simulation methods used concurrently (i.e., different parts of the UUT are simulated using different levels).

Figure 4 illustrates the relationships of these levels of simulation in a digital or hybrid UUT. The analog circuit level of simulation is needed when the simulated faults are involved with circuit components or when the measurements involve time constants which are close to the time constants inherent in the circuit. This level would be needed for measuring rise times, overshoots, and effects of switching spikes in some forms of digital circuits.

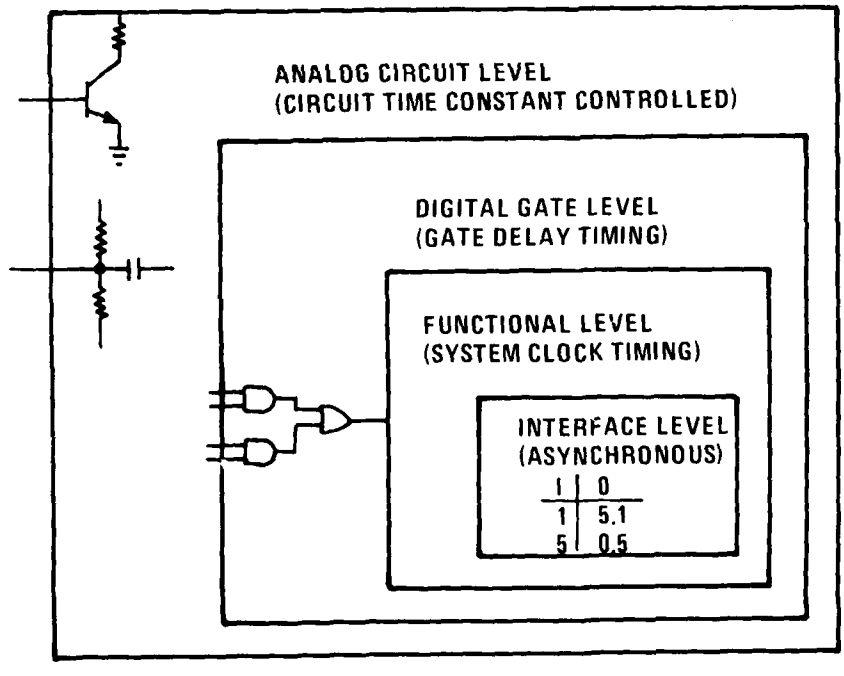


Figure 4. Levels of Simulation

The digital gate level of simulation is needed if the UUT contains a lot of digital logic or if internal digital faults are to be simulated in larger system functions. The simulation proceeds at about the gate delay level of time increments, considerably faster than the analog level simulation. Some forms of gate level simulation have enough fidelity to allow simulation timing analyses, such as in looking for race conditions; others do not. If multi-level simulation is supported in the UUT simulator, situations requiring greater detail than supportable at the gate level can be simulated at the circuit level.

The functional level of simulation involves dynamic input/output characteristics of subcomponents. In a digital system, the functional level would correspond to calculating only the outputs at the chip pins for every clock period. Signal details between these times and details of internal behavior are suppressed. Computation time is reduced but each assumed different internal fault would require another function implementation to simulate its effects.

The interface level of simulation is a special case of the functional level. Only static, typically table-driven, input/output functional relationships are assumed. New sets of inputs as well as different faults imply additional sets of entries in the input/output relationship table.

A second dimension of the UUT simulator attribute space is the method of interfacing the UUT simulator to the automatic test equipment system which is executing the test software to be verified. The nature of the interface has a strong impact on the simulator constraints, cost, and the extent to which the simulator can displace the need for the real UUT.

Figure 5 shows the three ranges of interfaces which we feel cover UUT simulator solutions of interest. The first, the analog signal interface, provides the most independent form of interface. It assumes that the actual interface adapter for the UUT will be used, and the real UUT is mimicked at the interface pin level. The solution requires A/D converters, level shifters, isolators, etc. as may be required to measure input signals, generate realistic responses, and isolate the simulator system from the test equipment.

Figure 6 gives a pictorial representation of what this simulator may look like and how it would be connected to the ATE system. Depending on the range of UUTs simulated, the simulator may have to sink much power or, conversely, generate it to drive the loads switched in on the test equipment side. To be anywhere near generic, this method of interfacing may require as much test equipment as the ATE system itself.

The realism of this form of interface is offset by the following negative aspects of the interface:

- Only low bandwidth UUTs may be capable of being simulated.
- High throughput simulation processing is needed to provide real-time responses. The instruments at the ATE side of the interface may require full speed responses to make the proper measurements.
- Signal speed is limited by the bandwidth of the A/D and D/A converters. Usually the simulation processing would saturate much earlier than the converters. But some forms of simulations

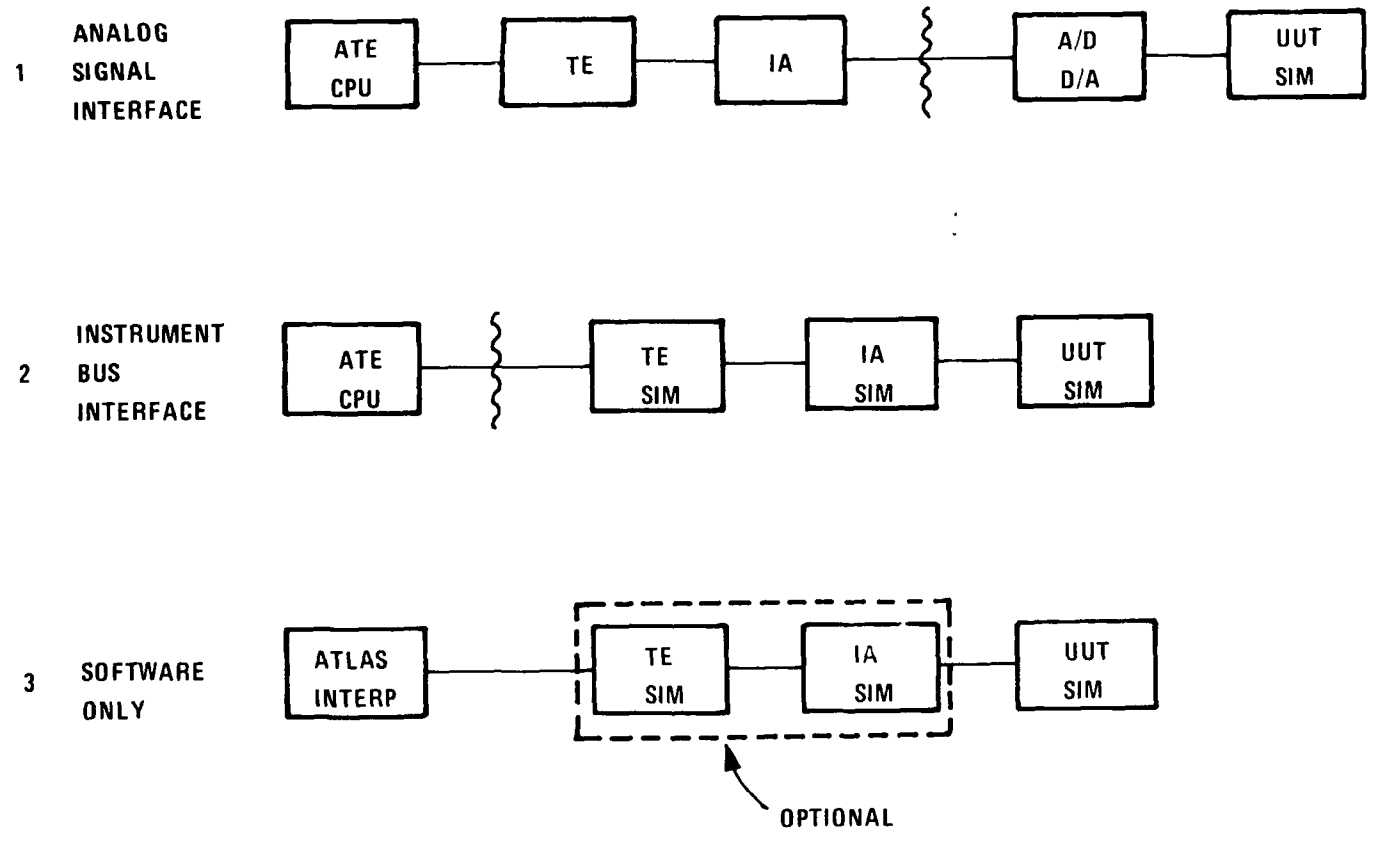


Figure 5. UUT Simulator Interface Methods

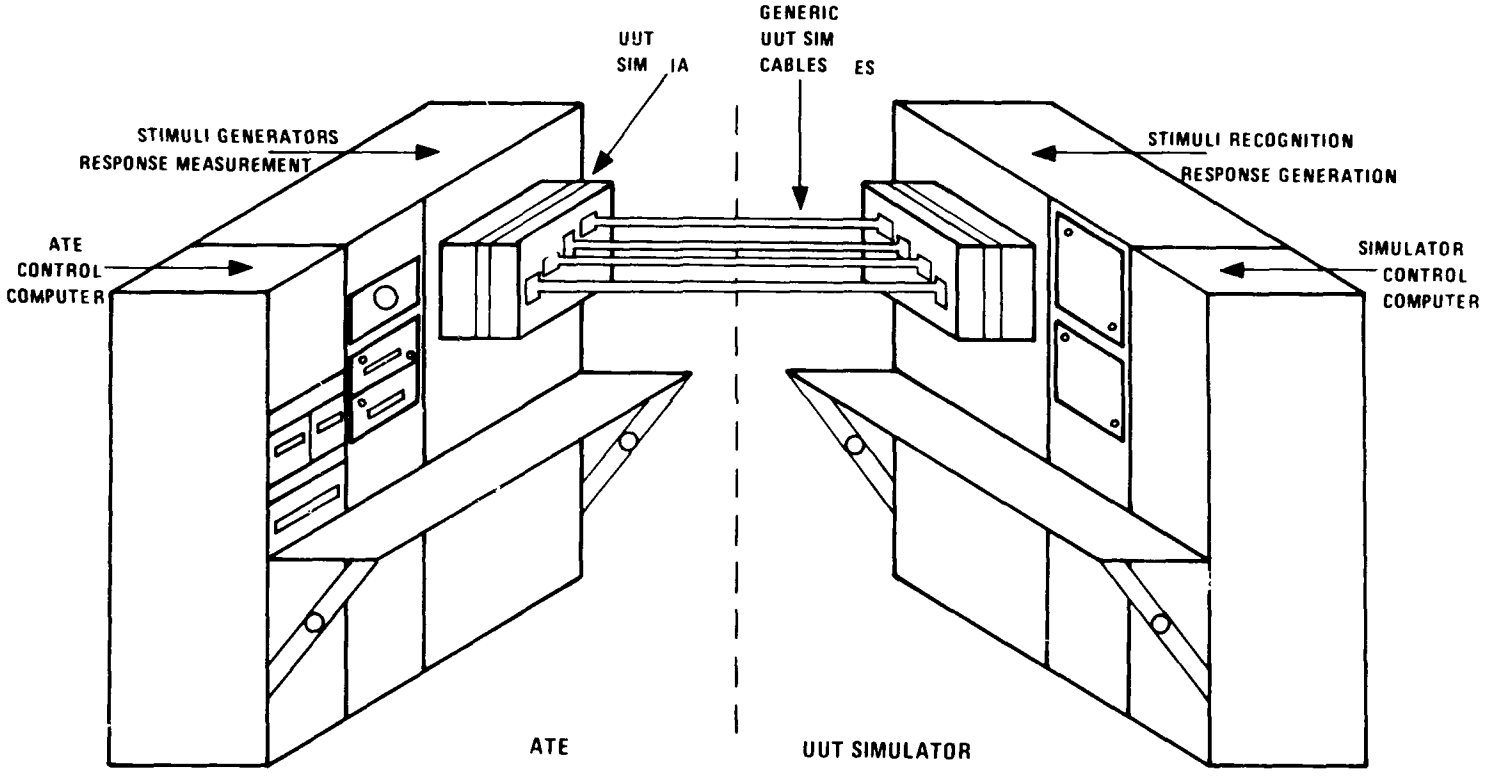


Figure 6. Drawing of UUT Simulator (Analog Level)

such as the interface level may produce waveforms fast enough to reach the speed limits of the converters.

- Extraneous noise and grounding problems may be introduced, which may detract from the test program verification activity. These problems may produce undesirable responses for test programs and the developers or validators may be forced into chasing problems which do not even exist in the real UUT.
- If the test program contains bugs which are potentially damaging to the real UUT, these same bugs are also potentially damaging to the equipment supporting the UUT simulator interface.

A side benefit of the analog signal interface is that it does allow the testing of the interface adapter before it is connected to the real UUT.

The second interface alternative is the instrument bus interface. In this case we assume that the ATE system is modular and has a well-defined bus which controls the test equipment instruments. The strategy is to connect the simulator system to the ATE CPU at this bus. In addition to the UUT simulator, this interface also requires the following:

- A simulator for the test equipment. This simulator would interpret commands from the instrument bus, generate the signals as commanded, and measure the simulated responses as requested. A functional simulator has to be developed for each piece of test equipment.
- A simulator for the interface adapter. This simulator would have to be developed to properly represent the routing and any potentially active functions in the real interface adapter. Only a functional level simulator would be needed.

Figure 7 shows a pictorial representation of the instrument bus interfaced solution. Note that neither the test equipment bay of the ATE or the special analog instruments of the simulator for the analog interface case are needed. The bus interface solution basically supports two computers talking to one another, with information represented at all stages in digital form. The criticality of the timing is also removed as compared to the analog interface case. The simulation can proceed at a slower than real-time rate. Timing constraints would be encountered only if the ATE test programs contain time-outs which cannot be deactivated or slowed down.

A third interface alternative is the software-only solution. In this case both the interpreter for the test program and the UUT simulator are executed in the same computer. There are now a number of additional options in handling the interface. The test equipment and the interface adapter could be simulated, as in the bus interface case, but at a less precise level. In addition, the semantics of the test program statements could be executed directly at the UUT pin level without regard to the specific test equipment used to generate the stimulus or the nature of the interface adapter. A consequence, though, is that no verification of the interface adapter is performed.

SOLUTION COMPONENTS

To further differentiate alternative UUT simulator solutions, we can look in more detail at specific components that would make up a UUT simulator. We will then find that some components are common among classes of solutions and some are unique to specific classes. By examining these

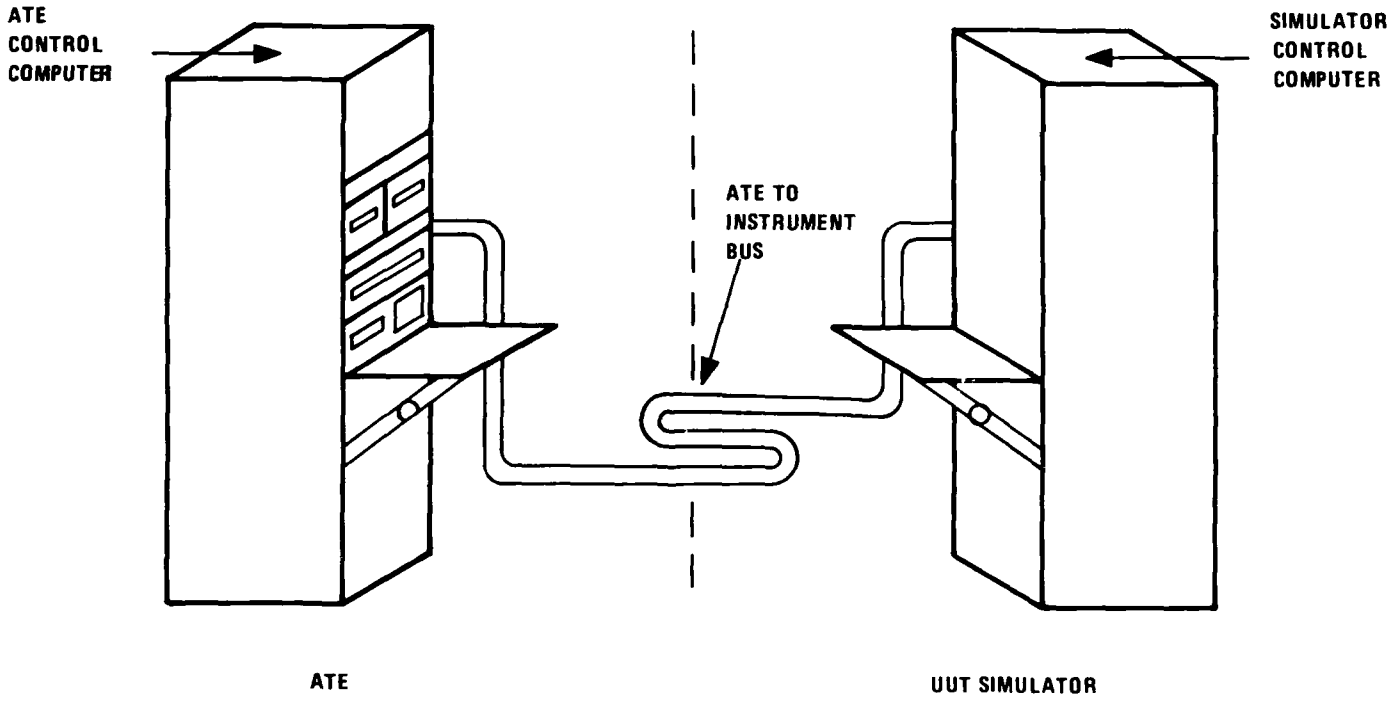


Figure 7. Drawing of Interface-Bus Visualization

components in more detail we can get a grasp on both development and usage costs as needed to perform trade-offs between solutions.

Figure 8 shows an overview of a signal flow model of a UUT simulator. By following signals through these components we can get a better feeling of the interaction of the components.

The input from the ATE system first passes through the set of functions labeled "Input Signal Recognizer," "Test Equipment Stimuli Generator," and and "Interface Adapter Input Simulator". The flow as shown corresponds to the bus interface set of solutions. For the analog interface solution only the signal recognizer (consisting of both hardware and software) would be required.

The signal then flows into the UUT simulation control function which has the job of passing it to the proper level of simulation. The figure shows the case in which all four levels of simulation are present. If only one level is supported, the routing is simplified.

The simulation of the UUT then performs transformations which ultimately result in simulated outputs. These outputs are then routed via the interface adapter output simulator, the test equipment measurement model, and the output signal generator back to the test equipment. This flow again corresponds to the bus interface solution. For the analog interface solution only the output signal generator would be required.

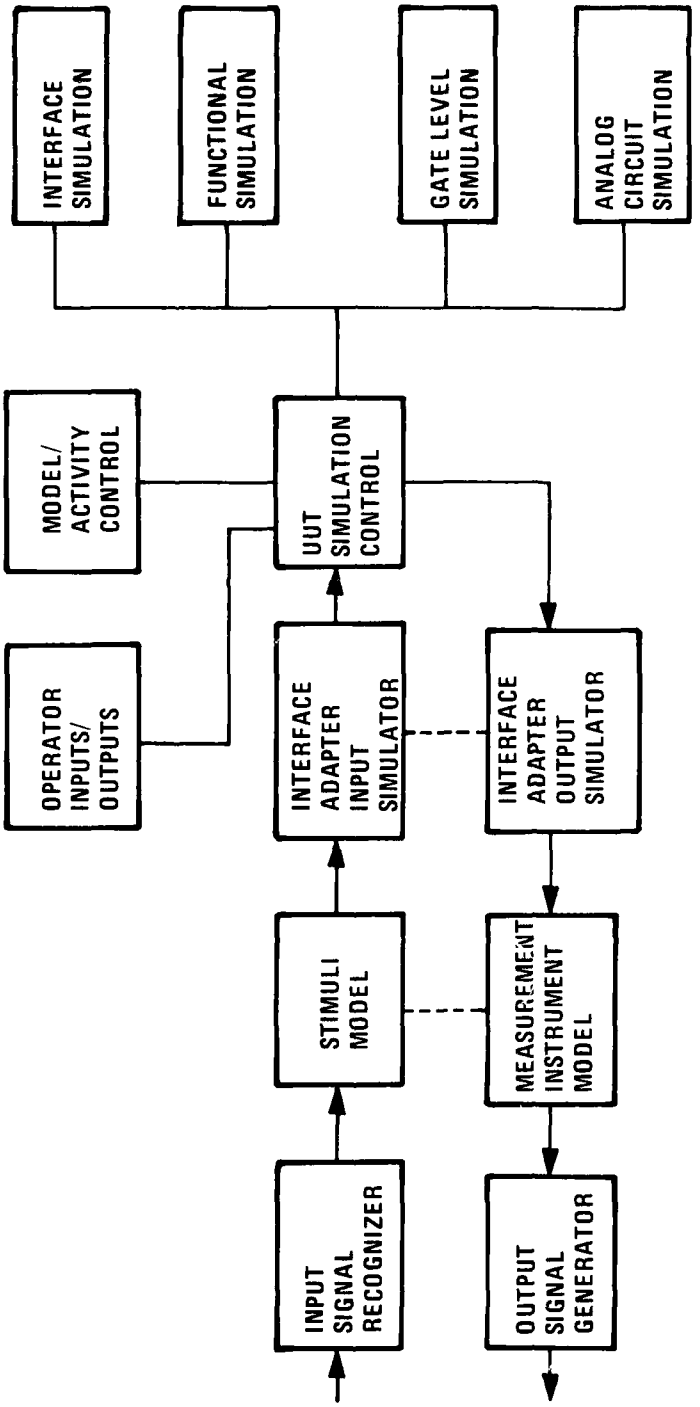


Figure 8. UUT Simulator Signal Flow Model

The two functions shown in the diagram that are not yet discussed are the Operator Input/Output and the Model/Activity Control. The operator input/output is required if the UUT has outputs that must be manually observed or inputs such as function switches. The model/activity control is used for controlling the simulation and for selecting model parameters and inserting faults.

Figure 9 shows a simpler signal flow diagram but emphasizing the software-only form of interfacing. In this case the ATLAS interpreter is needed in place of the test equipment and interface adapter simulators. A main point to be observed is that the right side of the diagram containing the simulations as well as the operator interaction is the same as in the previous diagram.

Figure 10 shows a hierarchical perspective of the components of a UUT simulator. The purpose of looking at these components in more detail is to allow us to assess their impacts on the various possible forms of UUT simulators with regard to development costs and usage costs. Comments on cost implications are given for most components.

Simulation Sequencing Control

This is the main controlling activity in the UUT simulator. It must ensure that the resources in the computer supporting the simulation are shared among the subcomponents of the simulator. It must also control the timely activation of components to ensure timing constraints are not violated.

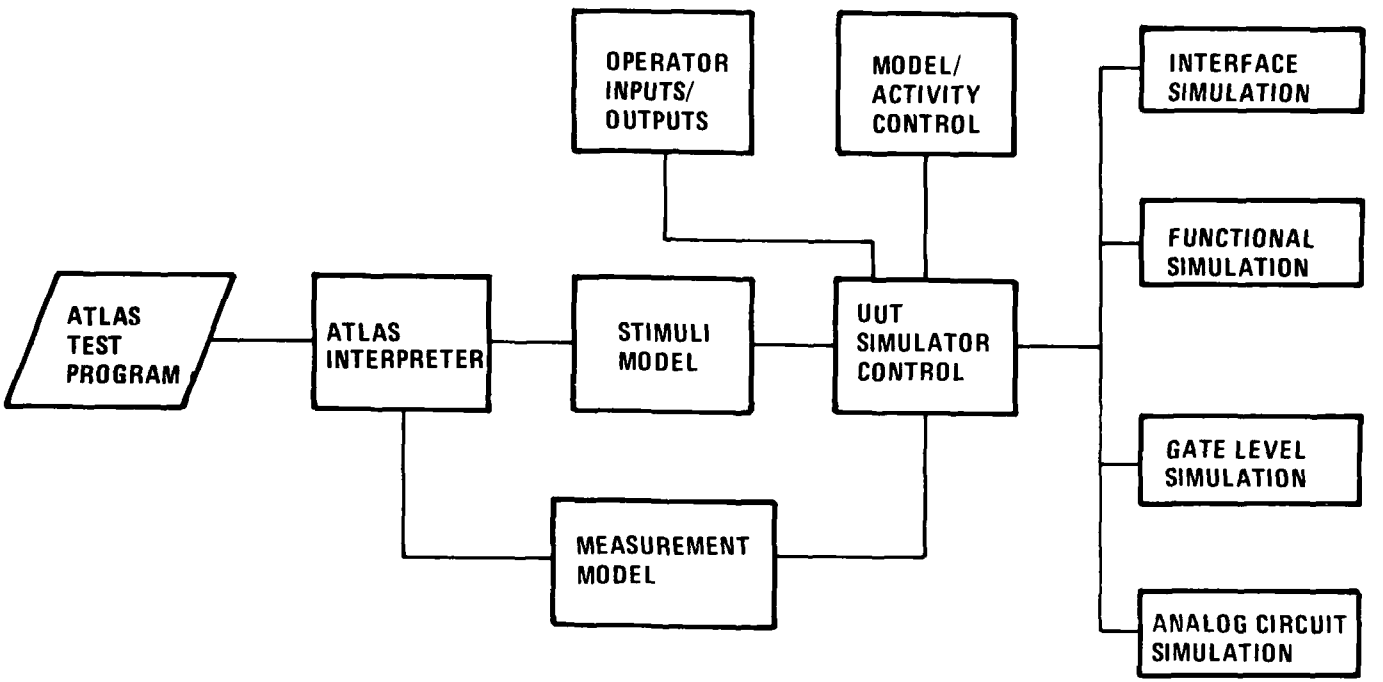


Figure 9. Software-Only Form of UUT Simulator

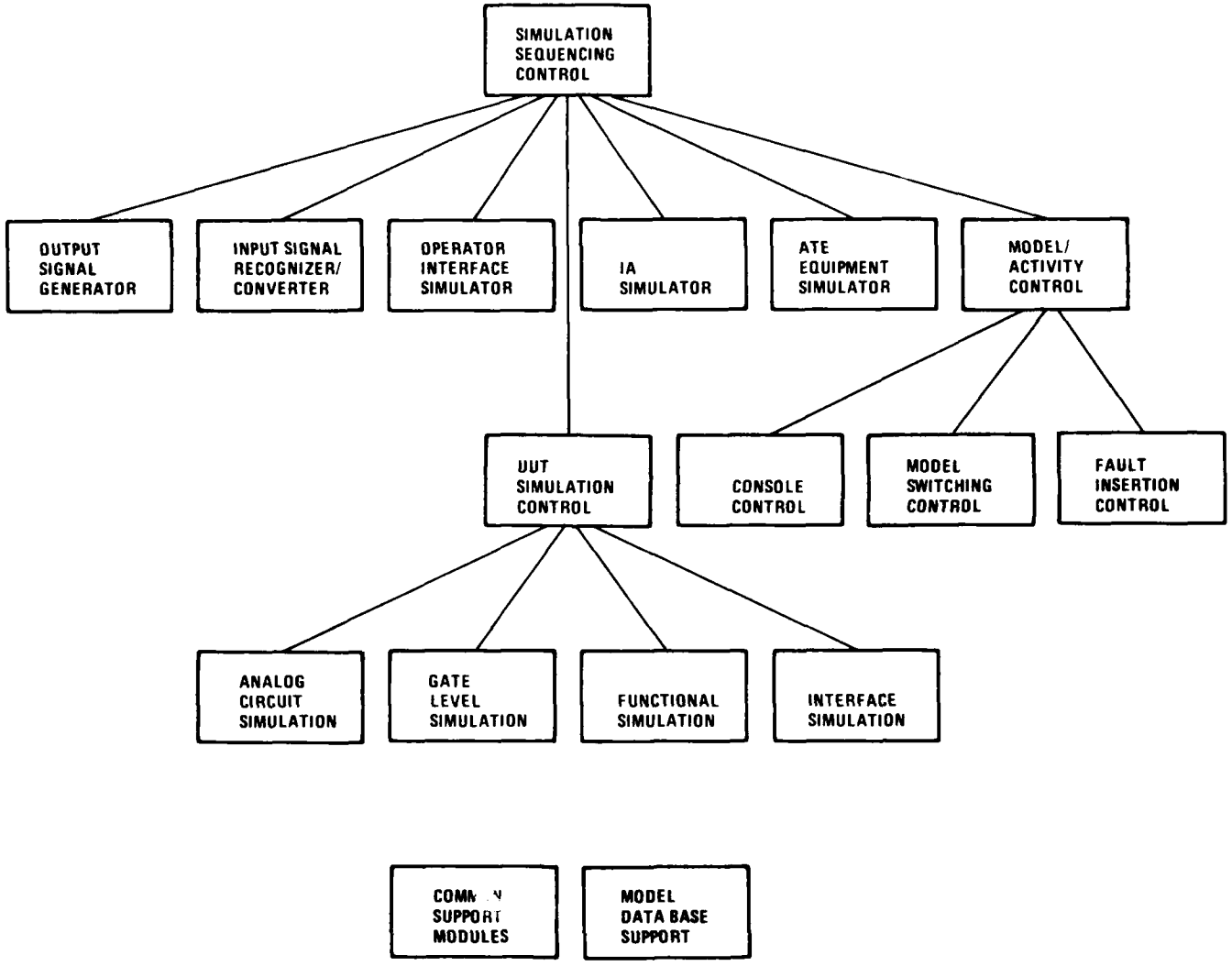


Figure 10. UUT Simulator Components

Development costs are similar for all simulation levels. The analog interface solutions require extra hardware support to meet stringent real-time constraints. The bus interface solutions require an intermediate level of complexity for this function, and the software-only interface solutions require only minimal support.

Usage costs are similar for all simulation levels. The analog interface solutions imply higher usage costs because of the more complex and higher real-time demands for controlling the sequencing.

UUT Simulation Control

This function controls the activation and coordination of the simulation models. It must support matching time increments when more than one level of simulation is used. It must also support interfacing signal levels between the different simulations.

For the simulation levels involving only one type of simulation, the control functions are relatively simple and the development costs should be low. For the multilevel simulation case, extensive control and interfacing are needed to realize the benefits of using all the simulation levels. The development costs are independent of the interface method.

Only the multilevel simulation case would require a significant usage cost. These costs should be more than compensated for through the gains in efficiency in performing the total simulation. The usage costs are independent of the interface method.

Analog Circuit Simulation

This component performs the analog simulations for all the parts of the UUT to be simulated at this level. It is assumed that some existing system for analog circuit simulation is either used directly or adapted to specific requirements for performing UUT simulations. The simulation is ideally compatible with forms of simulation used for the UUT during other phases of its development. For example, if analog simulation is used during design, it is desirable to adapt this simulation for use as a UUT simulator oriented to the support of Verification and Validation. The choice of which analog simulator to be used must be based on more specific system requirements and development contexts.

For the simulator solutions involving either the analog simulation level only or the multilevel simulation, the development costs are expected to be high. It can take as much as a few man-years to adapt an existing analog simulation system to a new computer environment.

Adapting an analog simulator for use as a UUT simulator (particularly one implemented on a minicomputer) would require both the adaptation to the new environment and the addition of support to facilitate topology and circuit component changes to simulate the insertion of faults. For the UUT simulator solutions not involving analog simulation, there are no associated development costs. The costs are also independent of interface methods. But if the analog interface is to be supported and high simulation speeds are required, much additional development is required to re-implement the analog simulation on higher speed processors (such as special microprogrammed processors or array processors).

For the simulator solutions involving only analog simulations, the usage costs are expected to very high. Either long periods of a conventional computer are required or special (and expensive) equipment must be procured and amortized during the use of the simulator. For the multi-level simulation solutions, costs are expected to be moderate because only parts of a UUT would have to be simulated at the analog level. The usage costs are again independent of the interface method unless the analog interface solutions require high-speed processing. In this case the simulation costs are expected to be very high.

Gate-Level Simulation

This component performs the gate level simulations for all parts of the UUT to be simulated at this level. It is again assumed that some existing gate level simulator (typically from a digital test program generator) is available for adaptation to the UUT simulator environment. Fewer internal modifications are likely needed because the fault insertion mechanisms are simpler and may already be built into the simulator.

In those simulator solutions where gate level simulation is needed, the development costs are assumed to be moderate. The assumption is that some existing simulator can be modified and tailored for the UUT sinulator environment. The development cost is independent of the interface method used as long as no extraordinary speedups are required to meet throughput rates higher than the original simulator was designed for.

Usage costs would tend to be moderate to high, depending on the amount of gate level simulation needed in specific simulations. The costs would probably range between the cost of doing more detailed analog simulations and the cost of doing functional level simulations.

Functional Simulations

The general support provided at this level of simulation is assumed to be minimal. At most, some primitives and low level functions supporting functional simulation would have to be provided a priori to the actual simulations. Therefore the one-time development costs are low. However, for each UUT to be simulated at this level, an extensive development may have to be undertaken to build up both the properly operating functional simulation and functions for simulating fault models.

Development costs are expected to be low.

The cost of running the simulated functions will be relatively low compared to gate level or analog simulation. However, a one-time development effort must be planned for each UUT. This is expected to be much more costly than entering the models and topologies of an analog or digital simulation.

Interface Level Simulation

It is assumed that this form of simulation would require some form of preprocessor or simple special purpose language to support entering input/output relationships to be simulated. Simple dynamic cases, such as sequences of signals, should also be supportable for the interface level of simulation to be effective.

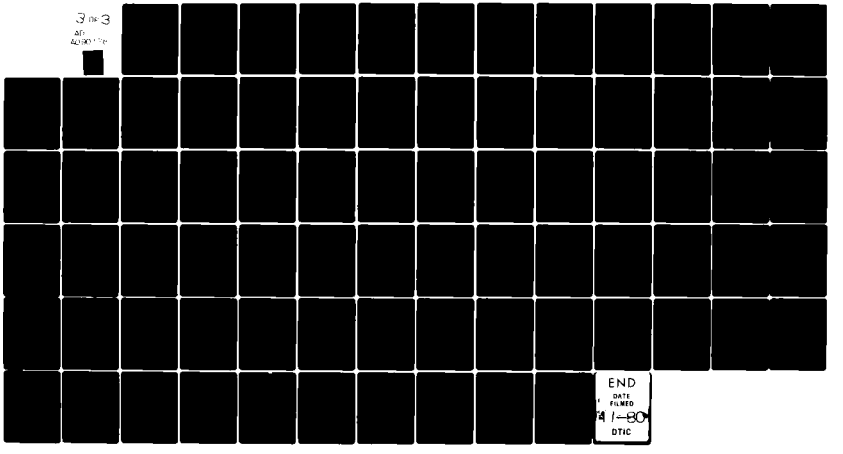
A low to moderate development cost would be incurred (as compared to an analog or digital gate simulation). For UUT simulator solutions containing only interface level simulation, the development should be more complete than if other forms of simulations are also supported. In the latter case, more difficult simulations can be done in more detailed simulations.

Usage costs are expected to be low for all cases.

Output Signal Generator

The function of this component is to match the output of the simulator to the input of the VTE system at the proper interface level. This component would be completely different for the different methods of interfacing. For the analog interface methods, both hardware and software along with real-time constraints are implied. Consequently, both the development costs and the usage costs are high. The software-only sets of solutions would be least costly because there are no real-time constraints, and the formats can be designed to be compatible. The instrument bus interface solutions require an intermediate level of complexity. The UUT simulator signals

AD-A090 178 HONEYWELL SYSTEMS AND RESEARCH CENTER MINNEAPOLIS MN F/G 14/2
UNIT UNDER TEST SIMULATOR FEASIBILITY STUDY.(U)
JUN 80 S J NUSPL, D T LEE, J P HUDSON F33615-79-C-1811
UNCLASSIFIED 80SRC46 AFWAL-TR-80-1091 NL



would already have been accepted and converted by the interface adapter and test equipment simulators. Therefore, the output signal generator has to output the response in the proper protocol structure required by the instrument bus.

The development costs are independent of the levels of simulation. As mentioned above, the analog interface would require costly hardware and software developments. The software-only solutions would imply the least cost.

The analog interface solutions would imply relatively high usage costs, considering higher maintenance and amortization of the additional hardware.

Input Signal Recognizer/Converter

The function of this component is to convert the signals from the ATE system into internal formats appropriate for the simulation inputs. The specific functionality of this component is strongly dependent on both the simulation technique and the interface method:

- For the analog interface method, the component consists of both hardware and software, implying high costs for both development and usage.
- For the bus interface method, this component consists of simple digital hardware and software to handle the instrument bus protocol. Both development and operation costs would be less.

- The software-only solution would be least complex and least costly.
- Both development and operation costs would be similar for all levels of simulation except the interface level simulation method. In this case the input recognizer would be required to recognize sets of input signals in order to generate the proper outputs. The implications are that both the development and operations costs for the input signal recognizer would be higher than for the other simulation techniques.

Operator Interface Simulator

The function of this component is to display any UUT outputs which the operator is expected to observe and to accept operator inputs (such as via soft keys) when requested by the test program. It is assumed adequate to represent the UUT outputs via alphanumerics on the UUT simulator console.

Development costs are similar for all simulator solutions.

Usage costs are insignificant.

Interface Adapter Simulator

This component is needed for the bus interface set of solutions. It must be developed specifically for each UUT, and it should be based on the design or schematics for the real Interface Adapter used.

Initial UUT simulator development costs are low.

A one-time development cost is incurred for each UUT, but once developed, the usage costs should be insignificant.

ATE Equipment Simulator

This component is needed in the solutions using a bus interface. The simulator is driven by commands received by the input recognizer and simulates stimuli generation and performance of measurements. The initial UUT simulator development would require the development of simulators for a basic set of test equipment functions and instruments. Additional instrument simulations must be developed as new instruments are used for new UUTs. However, functional simulations should be adequate.

Initial UUT simulator development costs depend on the complement of instruments assumed supported by the ATE.

After a sufficient number of UUT simulators have been developed using the same system, new instrument simulations should have to be added only infrequently. The run-time usage costs should be insignificant.

Model/Activity Control

The function of this component is to accept user inputs to control the progress of the UUT simulation. It is shown as the top level control for the following three lower level functions.

Console Control--This component processes console input commands and supports the output of information to the console. Examples of lower level functions supported are command decoding and format conversions. User support functions needed to debug UUT simulators during their development (such as traces, dumps, and breakpoints) must also be provided.

Development costs are independent of the interface method or level of simulation.

Usage costs are negligible.

Model Switching Control--This component is needed for only the multi-level simulation solutions. Its function is to control the switching in and out of the different simulation models in order to support faster total simulation of time. Part of the function of this component also consists of supporting the interfacing of different UUT component models when transitions in levels are made.

Moderate development costs can be assumed for the multi-level simulation solutions.

Usage costs are negligible.

Fault Isolation Control--This component supports the control over inserting faults or changing models to produce the symptoms of a fault. Semi-automatic control is assumed to help expedite cycling through the process of starting a test, inserting a fault, watching the ATE system react, and

resetting the UUT simulator to prepare for insertion of the next fault. This is one of the key support functions in the UUT simulator system for aiding the independent V&V process.

Costs are dependent on specific simulation techniques.

Usage costs are dependent on simulation techniques and are comparatively low.

Common Support Modules

This component consists of a collection of low level functions needed by many of the higher level UUT simulator environment modules. A reason to separate them is to maximize commonality and therefore reduce both development costs and later maintenance costs.

Development costs are low to moderate.

Usage costs are low.

Model Data Base Support

This component supports information structures needed for data-base-driven simulations such as an analog simulation and a gate level simulation. The ability to separate this module out as a separate component depends on the specific simulation techniques chosen and how the simulators were developed.

Other Support Components

In addition to the UUT simulator components required during the actual running of a simulation, another set of support components is needed to prepare for the execution of the simulation. These components are partitioned and illustrated in Figure 11.

The main cost associated with these components is development costs. The usage costs can be considered negligible. A brief description of each of the components is given below:

Interaction Model Builder— This is a postulated component that takes a source description of the UUT to be simulated and builds a data base describing how the UUT components interact. It would perform consistency checks as the data base is built. The nature of this component is again very dependent on the forms of simulations used in the UUT simulator systems.

Interface Model Preprocessor—It is assumed that the interface level of simulation would have to be supported by only a simple preprocessor or at most a simple, special purpose language translator. The function of this preprocessor is to accept the input/output descriptions comprising the UUT component models and to build the internal table in a form compatible with processing software.

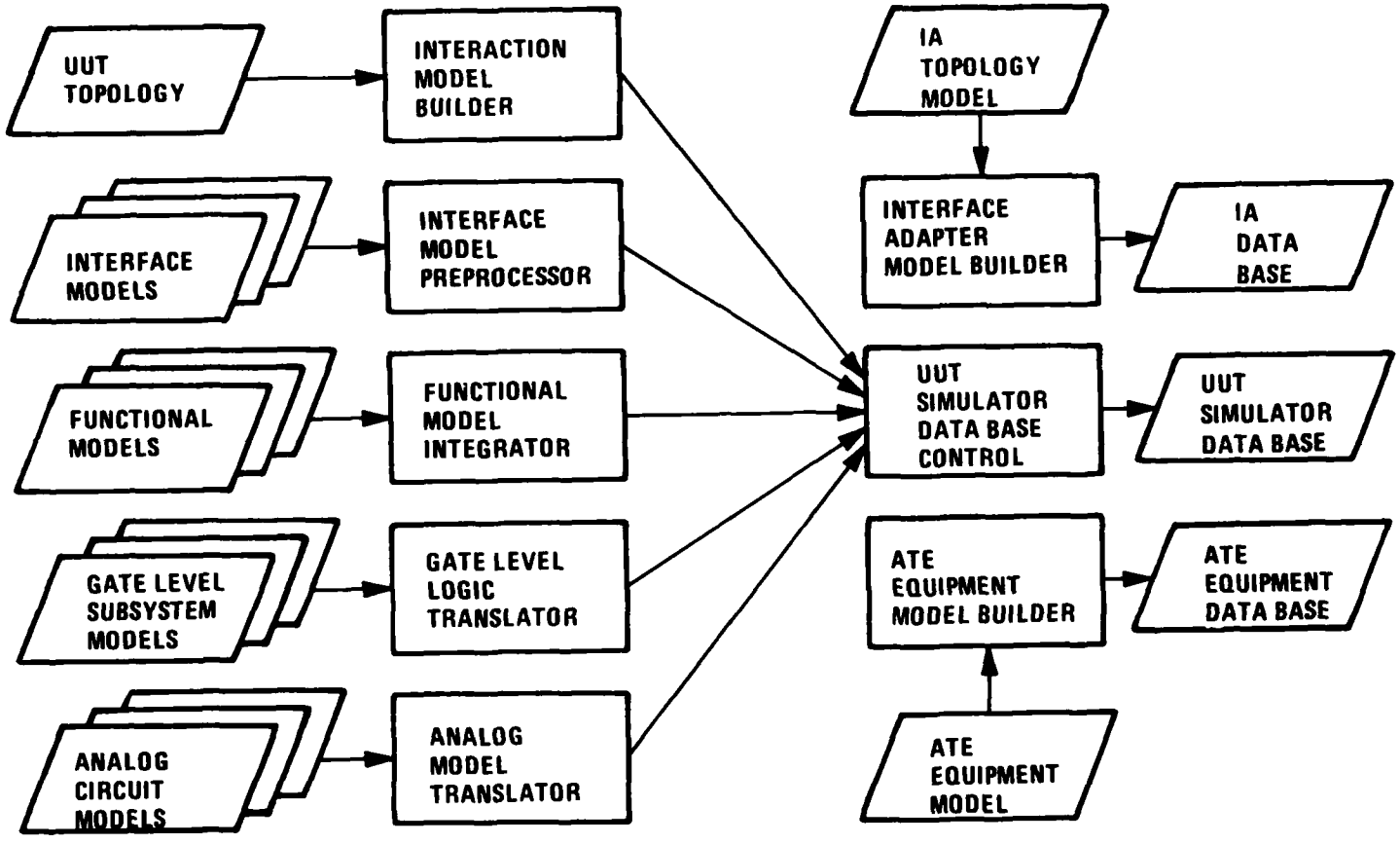


Figure 11. UUT Simulator Support Functionality

Functional Model Integrator—This component supports the coalescing or binding of independently developed modules. The most rudimentary version of this model is simply the link editor of the host computer support system. In this case the functional modules are assumed programmed in a high order language supported by the host.

Gate Level Logic Translator—If this form of simulation is used in the UUT simulator, this component performs the translation of register transfer or gate level circuit descriptions into a form compatible with the internal simulation data base. This translator would normally be provided by the gate level simulation system incorporated into the UUT simulator system.

Analog Model Translator—This component would be typically considered as part of the analog simulation system. It accepts descriptions of analog circuit models, performs checks, and builds the data base for the analog simulated sections of the UUT.

Interface Adapter Model Builder—If the simulation system requires models for the interface adapter, some form of preprocessor support is needed. Because the models must be functional at most, the preprocessing should be simple.

UUT Simulator Data Base Control—This component provides common data-base construction support and access to the data base, and performs checks as well as data compression. Ideally the data base control is common for all the simulation techniques. The ability to accomplish this and the structure of this component depend largely on the specific simulation approaches and implementations chosen.

ATE Equipment Model Builder—If the simulation system requires models for the ATE equipment, some form of support is needed to develop the internal data base models. The models need again to be at only the functional level, and therefore the support required for this function is assumed to be minimal.

Support Functionality Control—This component (not shown on Figure 11) includes high level control of the UUT simulator support functionality. It includes functions to allow user interfacing, to perform checks, and to control activation of the specific lower level functions.

SOLUTION SELECTION

The set of UUT simulator solutions covered by the component descriptions discussed above covers a very wide range. To zero in on the most cost-effective solution an appropriately scoped trade-off was performed. The initial direction of this study was to determine the relative benefits and costs in terms of a solution matrix as shown in Figure 12.

The cost factors which were considered included the following:

- UUT Simulator System Development Cost: A one-time development cost which is proportional to the complexity of the components required for the UUT simulator. Comments about the relative complexity of the UUT simulator components were given above in the discussions of the components.

LEVEL OF SIMULATION	C SOFTWARE ONLY	B INSTRUMENT BUS INTERFACE	A ANALOG SIGNAL INTERFACES
1 ANALOG			
2 GATE LEVEL DIGITAL			
3 FUNCTIONAL			
4 INTERFACE			
5 MULTI-LEVEL			

Figure 12. Initial Trade-off Matrix Format

- Incremental UUT Simulator Usage Costs: The assumption in these costs is that the UUT simulator system has been developed and is stable. The usage costs now include the procurement and amoritization of additional copies of the simulator system, developing specific UUT simulators, and usage costs while the simulator is used during V&V.

To arrive at estimates of the UUT simulator component costs, the following development and deployment scenario was assumed:

- A generic UUT simulator system environment is initially developed to conveniently support the later addition of specific

UUT simulators. This system can be considered analogous to a generic ATE system that is developed with general support capability so that specific instruments and test program sets can later be added.

- The deployment of a specific UUT simulator system requires that a new copy of the support hardware be procured and installed at the place in which it is to be used. It is assumed that the cost of this additional copy of the system must be amoritized over the life of the system.
- When a specific simulator for a UUT is to be added, some one-time development costs are incurred for that UUT, depending on the method of interfacing and the level of simulation.
- During the use of the UUT simulator in performing V&V, both the UUT simulator equipment and the test equipment which must be tied up while performing the task must be taken into account in determining the incremental usage costs.

While the cost factors were being developed for the entries in the trade-off matrix, it was found that most factors were dependent on either the simulation level or the interface method. Consequently, it was decided to reduce the number of entries in the trade-off by separating the simulation levels and the interface methods.

Table 22 contains the development cost estimates for the UUT simulator components. The numbers entered represent estimates of the number of statements in the source code required to implement the component or

TABLE 22. COMPONENT DEVELOPMENT COSTS

Major Components		Base Line	Simulation*					Interface*		
			1	2	3	4	5	C	B	A
1.0	Simulation Sequencing Control	1,000						-50		+400
1.1	UUT Simulation Control	2,000					+500			
1.1.1	Analog Circuit Simulator (adapt)	10,000		-100	-100	-100				
1.1.2	Gate Level Simulator (adapt)	5,000	-100		-100	-100				
1.1.3	Functional Simulator (interface)	5,000	-100	-100		-100				
1.1.4	Interface Simulator	3,000	-100	-100	-100		-25			
1.2	Output Signal Generator	1,000						-50		>500
1.3	Input Signal Recognizer	2,000	-75	-75				-15		>500
1.4	Operator Interface Simulator	2,000								
1.5	IA Simulator (or Atlas testers)	2,000						+425		>1000
1.6	ATE Equipment Simulator	4,000	-100	-100	-100	-25		-50		-100
1.7	Model/Activity Control	2,000					+50			
1.7.1	Console Control	2,000								
1.7.2	Model Switching Control	2,000	-75	-75	-75	-75				
1.7.3	Fault Isolation Control	3,000	-25	-50						
1.8	Common Support Modules	4,000				+20				
1.9	Model Data Base Support	2,000			-75	-75	+25			
2.0	Support Functional Control	500								
2.1	UUT Simulator Data Base Control	4,000	-50	-50	-50	-50				
2.2	Interaction Model Builders	4,000	-75	-75	-75	-75				
2.3.1	Analog Model Transistor	5,000		-100	-100	-100				
2.3.2	RTI Translator	1,000	-100		-100	-100				
2.3.3	Functional Model Interface	1,000	-100	-100		-100				
2.3.4	Interface to Top Integrator	1,000	-100	-100	-100					
2.4	IA Model Builder	1,000						-100		-100
2.5	ATE Equipment Model Builder	1,000						-100		-100
	Total	70,500	41,750	32,000	33,500	34,500	80,050	71,500	70,500	>97,500

*Numbers correspond to deviations from baseline

equivalent source code statements if hardware is involved. These numbers represent a rough order of magnitude estimates based on the authors' perception of the component and on experience in working with or developing similar functionality components. The objective of this exercise was to get a first cut at providing inputs for the trade-off study. These figures of the number of statements were converted to dollars by assuming a $50 per statement cost of software development.

Table 23 contains a break-out of the cost components factored into the incremental UUT simulator usage costs. There was considerable difficulty in coming up with these estimates. First, the development activity required to develop a TPS varies widely, depending on the complexity of the UUT. Second, there is no consistent information on costs of UUT development in the open literature.

To get some comparative numbers for the UUT simulator solution alternatives, the following assumptions and estimates were made:

- A nine-man-month TPS development size was assumed.
- It was assumed that some UUT associated development costs are incurred to make the simulator operational. These costs include both entering the UUT description and preparing auxiliary simulators (such as for a UUT adapter or special instrument).
- A rough estimate was made for each of the simulation levels and interface methods, using a figure of $7K per man-month to translate the cost into dollars.

TABLE 23. INCREMENTAL USAGE COST FACTORS

	1	2	3	4	5	C	B	A
UUT Associated Development Costs	24.5	10.5	59.5	31.5	31.5	33.5	38.5	45.5
UUT Simulator Equipment Usage Costs	47.0	13.5	9.0	16.0	30.0	30(5)	30(5)	162.0
ATE System Usage Costs	16.0	16.0	16.0	16.0	16.0	16.0	16.0	162.0
	87.5	40.0	84.5	63.5	77.5	79.5	84.5	369.5

(Assume 9 mm TPS Development Size)

- Usage costs of the UUT simulator were estimated, based on a percentage of time the UUT simulator would be tied up and on the amount of equipment tied up in performing the simulations. A basic charge of $45 per hour of UUT simulator CPU time was assumed.

- ATE system usage costs were based on the assumption that only the CPU portion of the ATE would be tied up for all solutions except those involving the analog interface method. In the latter case it was assumed that additional ATE equipment tied up in using the simulator was about an order of magnitude more costly than the CPU alone. This reflects the trend that CPU costs are still going down, whereas instrument complexity and costs are going up.

Figure 13 is obtained when the results of these costing exercises are plotted.

- The analog interface-based solutions are the most expensive to develop and to use. Because there is not a proportional payoff, this interface is not recommended for use in a UUT simulator system.

- There is not a clear cost advantage for either the software-only or the instrument bus interface-based solutions. Factors other than cost must be used to determine which is optimum for specific cases.

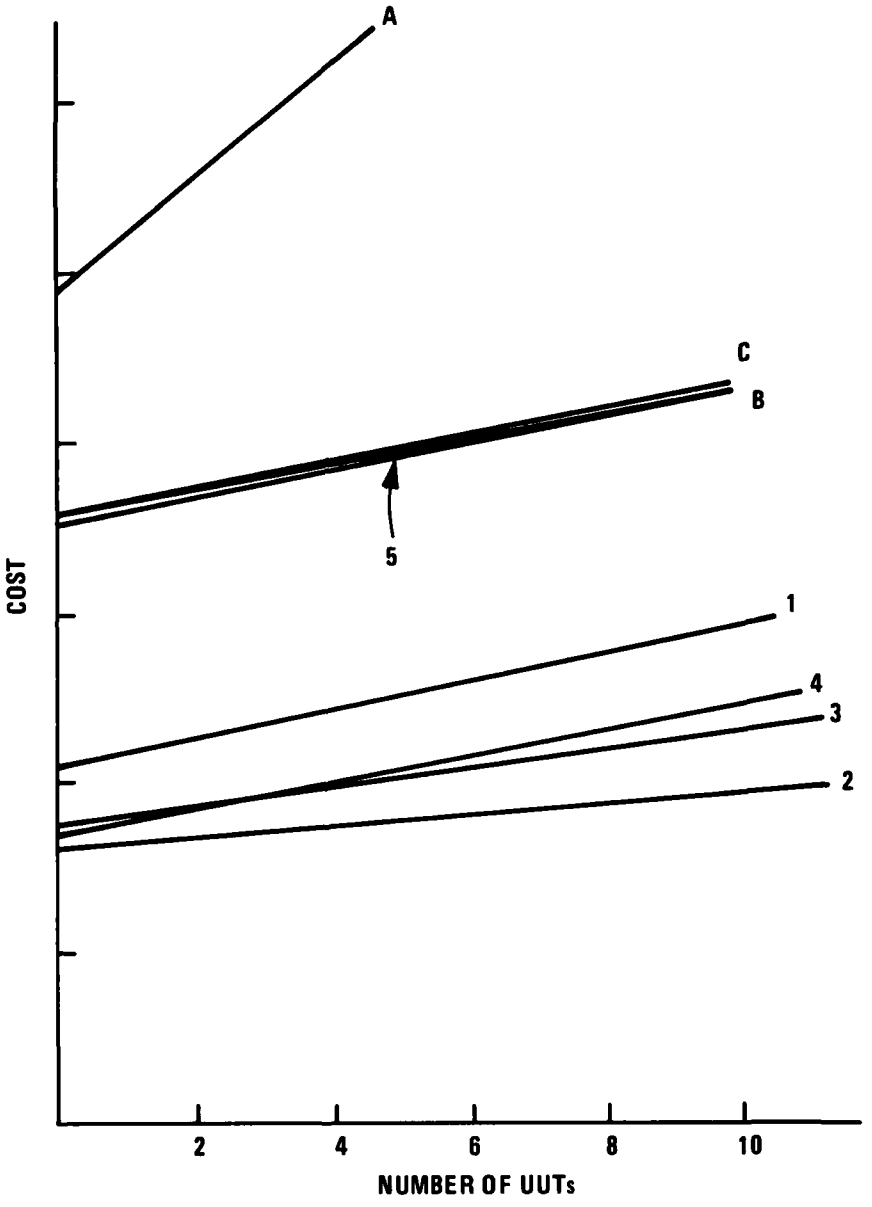


Figure 13. Summary Costs for UUT Simulator

The following are additional minor conclusions:

- If a family of UUTs can be adequately served by one level of simulation, a UUT simulator system based on this simulation only can become a cost-effective UUT simulator system. The primary saving is development costs.
- A major part of the initial development costs are independent of the simulation technique.

SECTION 5

SOLUTION DISCUSSION

This section provides more comments on the UUT simulator. The following specific areas are presented:

- Solution Recommendations and Discussions: More details of the recommendations are presented and implications are discussed.
- Usage Scenarios: A few example scenarios of how the UUT simulator would be used are outlined.
- Additional Usages: The UUT simulator can be used potentially in areas other than just supporting V&V. Some of these areas are described below.

SOLUTION RECOMMENDATIONS AND COMMENTS

Our recommendation is to ultimately migrate to the multilevel UUT simulator solution. In the previous section it was seen that the multilevel solutions had a high development cost associated with them. If over a period of time UUT simulators are developed to use individual simulation levels but of different types, the cumulative development costs would not be any lower. Unless such individual developments are closely coordinated, the result will not be as unified as if a coherent approach to a multilevel simulation solution were started early in the UUT simulator development.

Perhaps the most important factor is the existence of a unified approach from the start; different levels of simulation can then be brought into the system as they are needed.

It was seen in the previous section that on a cost basis the software-only and bus interface solutions are about equal. The specific choice between these two depends on the nature of the ATE system which the UUT simulator has to support. The following are the two main cases:

- If the ATE system is strongly based on a standard ATLAS compiler and if it can be guaranteed that no "idiosyncracies" of the compiler or ATE system get in the way of having a standard, reliable, and controlled interface to the UUT simulator, the software-only solution is good.
- If the ATE system is based on a well-defined bus structure connecting the Central Processor of the ATE system to the stimuli generation and measurement instruments, the UUT simulator may be best interfaced to this bus. The clear advantage is that the simulator interface does not depend on the source language that the test program is written in. For example, ATLAS may be the required standard, but to improve performance of a specific test, a TPS developer may request and receive a waiver to use another language in a minor part of the system. Such a waiver would have a severe negative impact on the software-only solution. The bus interface would still be valid.

A key aspect of the decision concerning which interface to use is how well-defined the interface is and how well it can be controlled by the UUT simulator developer.

Another factor in the software-only vs bus interface decision is the relative timing of the interface definition and its implementation. Because it is desirable to start definition of the UUT simulator early, the choice may have to depend on which of the two is available first. For example, if the software interface can be well-defined for the MATE program early, then the software-only solution should perhaps be favored.

Like other new technologies, the use of a UUT simulator does imply some risk factors, particularly during the early usages. The following are some cautions to be observed to prevent less than desirable early usage experiences from preventing achievement of benefits when concept is more mature:

- A UUT simulator learning curve will apply during the early UUTs simulated on a generic UUT simulator. Because start-up and training problems may be encountered during the implementation of the early UUT simulators, the apparent implementation time may appear too long or the net cost-effectiveness may appear poor.
- TPS development costs may actually appear higher than without the use of the UUT simulator. As seen previously, some reasons are the cost of developing, modifying, and maintaining the simulator. In addition, the existence of the simulator will provide deeper insights into what the test programs are and are not doing. The process of resolving these differences will take more time than without the simulator. There is also the possibility that errors in the UUT simulator will cause some such problems; these would have to be treated like the design problems in the UUT itself.

In order to provide answers to the questions and issues raised above for the MATE program, a UUT simulator demonstration program should be initiated. The following are specific justifications:

- For reasons other than development costs, the concept of a UUT simulator needs to be pursued. The following are some of these reasons:
  - —Unavailability of UUTs. If a UUT is not available in a development that is on the critical path of a weapons program, there may be cost impacts outside of TPS development. A UUT simulator can help reduce this problem.
  - —Long term payoffs: Reduced LRU downtime, reduced maintenance time and ultimately reduced testing time will have larger system cost benefits than cost figures associated with just the TPS development.
  - —Schedule compression of TPS development: Once the UUT simulator concept is mature and supported at an adequate level, the UUT simulator will help to reduce the amount of time to develop a TPS and verify its correctness.
  - —Maintenance advantages: There are additional advantages which can be realized later during the maintenance period of a TPS. A UUT simulator can provide support for revalidation of a TPS either to correct a TPS problem or to accommodate an engineering change in the UUT itself, which force a change in the test program. A UUT simulator, if maintained during the lifetime of the TPS will also support the resolution of unanticipated

problems in the field by supporting contingency analyses. Analysis of what happened in catastrophic failures may be another benefit.

- The cost-effectiveness of a generic UUT simulator is not yet clear. The demonstration program can address some of the key questions on the cost implications to implement and maintain additional UUT simulators. Such experiments could be run after a demonstration UUT simulator system is available.
- The relationship of a UUT simulator to Automatic Test Program Generation is not clear. The indications are that the UUT simulator can be complementary to ATPG, covering V&V support in those areas where test components must still be generated manually.

USAGE SCENARIOS

The purpose of this section is to present some sample scenarios in which a UUT simulator is used. The following scenarios are described:

1. Minimal Baseline Scenario
2. Interactive Scenario
3. Batch Mode Scenario
4. Model Changing Scenario

Minimal Baseline Scenario

- Prior to the development of the specific UUT simulator, it is assumed that a generic simulator that supports the required functionality is available.
- The UUT is assumed already developed.
- The development of the specific UUT simulator is assumed started prior to the start of the ATLAS test program development so that it is available when test program V&V starts.
- First the initial version of the UUT simulator is built and thoroughly checked for consistency.
- The developed ATLAS test programs are then played against the simulator with careful control over the execution.
  - —Single step runs are used to check on interface activity and test program actions.
  - —Recovery points are retained to facilitate restarts after test program errors are encountered.
  - —Detected errors in the test programs are incrementally corrected and re-verified.
- A small set of simulated faults are selected and built incrementally into the UUT simulator model.
- Recovery is made at a point before a simulated fault is to be exercised in both the UUT simulator and in the ATE system.
- Fault simulation is activated, and the action of the ATLAS test program and the UUT simulator are recorded and later analyzed for proper operation.

- Further faults are activated, one at a time, to check test program logic and to verify proper operator messages and expected operator actions.
- This process is repeated by incrementally refining and increasing the fault modeling in the UUT simulator and by playing further tests against the simulator.
- The procedures and UUT simulator parameter selections used to force specific faults are also stored for later reenactment of the executions.
- When the performance of the test program is adequate, via use of the UUT simulator, verification is continued on the real UUT, using the real UUT Interface Adapter.
- Discrepancies in the operation of the test program against the real UUT as compared to the UUT simulator are analyzed. Those discrepancies requiring only simple changes in the UUT simulator are used to update the simulator for future revalidation of the test program after changes have been made. Those discrepancies which imply extensive changes in the UUT simulator need to be evaluated for cost-effectiveness. It may be better to leave the simulator deficient and require that the specific test be revalidated in the future on a real UUT.

Interactive Scenario

- Assume that the baseline scenario holds. The interactive scenario is an example of a detailed interactive activity in the middle of the baseline scenario.

- Suppose that an inconsistency is discovered in a signal applied at some UUT pins versus what the UUT simulator has programmed into it as being acceptable.
- The simulation is temporarily suspended by the developer.
- History of the simulation steps leading to the condition is printed on-line. This history would include activities in the interface between the ATE and the interface adapter and between the interface adapter and the UUT.
- The developer examines the test program and the corresponding trace output.
- If the information is not enough, recovery is made to a previous restart point, and more detailed tracing is requested.
- When found, the offending program step is corrected and the test program is re-compiled.
- A recovery point is selected and the test program execution is restarted.
- Trace levels are set high to provide output to ensure that the correction was made properly.
- If correct, the change to the source is inserted permanently.
- The session is continued.

Batch Mode Scenario

- Assume that the baseline scenario holds. This scenario is an example of how the iterative planning, preparation, testing, and analysis would be done under the constraints of a batch mode operation of the UUT simulator.
- The UUT simulator is assumed entered, and the ATLAS test program segments to be verified have been designed and coded.
- The steps to be verified are selected and batched into independent groups.
- The faults to be exercised are selected, and a plan for the batch runs is made.
- Trace levels are preplanned to get the proper amount of information from the simulation runs.
- The input is prepared for one batch run and submitted.
- While waiting for outputs, inputs for testing other batch groups are prepared, and results of previous runs are analyzed.
- When the results from the batch run are back, they are analyzed for expected responses and for errors.
- If no errors appear, further test groups are prepared.
- If there are errors, reasons for the errors are analyzed.
- If the reasons cannot be found after a reasonable analysis time, the batch is submitted for rerun, but with more detailed traces to support isolation of the cause of the error.

- The above cycle is repeated continuously with two to three batch groups in process concurrently to prevent idle time while waiting for output from the batch runs.

Model Changing Scenario

- Assume that the UUT simulator has multiple levels of simulation.
- Assume a normal session has been started.
- Suppose that a specific response from the simulator doesn't look good enough; for example, a better output waveform is needed.
- The session is stopped and set to restart at a previous recovery point.
- Interactive control over the simulator is exercised to select a more detailed model for the required section of the UUT.
- Simulation is continued, and the output is monitored carefully to insure that the improved response has been realized. However, execution is probably slower.
- If results are OK, the procedural steps to activate the more detailed simulation are recorded for replay during future revalidations of the same section of test program.
- The model is restored to the less detailed but faster version, and the simulation session is continued.

ADDITIONAL USAGES

During the conduct of this study most of the emphasis was placed on having the UUT simulator support V&V and TPS development activities. Given that one has a UUT simulator available, there is a potential set of additional uses aside from the mainstream use. The discussion below addresses some of these potential uses.

Semiautomated Support for Test Program Generation

While discussing the relationship of a UUT simulator to ATPG, we determined that a very good use of a circuit level form of simulator would be to actually support the semiautomated generation of test programs. The developer would perform the analysis to determine what stimuli should be applied, cause them to be applied to the UUT simulator, and watch the response. The response could also be captured and later used for signature recognition. Such a use of a UUT simulator would be complementary to some forms of analog ATPG and is useful when AATPG is not yet available or determined for a specific case not to be cost-effective.

Training Support

ATE maintenance and operator training are a large area of concern in the Air Force. A UUT simulator would be capable of supporting some of the more advanced aspects of maintenance and operator training. Some forms of training in which the exact operator or technician interface is

required (i.e., realistic instrument panels and controls) would not be appropriate for use with a UUT simulator as required for V&V. However, some of the more advanced training in which the student can be expected to interpret results from a CRT console are potentially supportable. For specific training exercises in which the UUT simulator would only be required to keep the ATE system cycling (i.e., all student interaction is with the ATE system), the UUT simulator could also be useful.

UUT Specification in TRD

There is currently some movement toward the use of ATLAS as a means of specifying test requirements more precisely in TRDs. One step beyond this philosophy is to use a UUT simulator that is defined in a concise and unambiguous language along with a precise description of the tests to be performed. The concept of a UUT simulator as well as much support would have to be matured to a high degree before this is possible.

Integrated Repository of UUT Behavior in Field

The UUT simulator can be used as a continually updated storehouse of information on a specific UUT. As failures as well as good experiences are obtained from the field, the UUT simulator can be updated. The resulting accumulated information can lager be used as a basis for decisions on LRU redesign or on rework of tests to correct for poor field performance.

Support for Post Mortem of Catastrophic Failures

If catastrophic failures occur because of factors which were not even entered into the design of the LRU, the UUT simulator may be used as a means of postulating what might have happened and what could be done to prevent a recurrence of the failure.

Interface Adapter Verification

Adapter verification consists of verification of the following:

- Continuity from the ATE to the desired UUT test point
- Isolation between UUT test points
- Insulation breakdown voltage measurements
- Functionality of adapter active circuitry or fixtures
- Transmission line characteristics
- Noise immunity

Verification of transmission line parameters and noise immunity is done only when problems of this type are anticipated or are actually experienced. This puts these considerations in the category of exceptional tests, generally requiring special equipment and fixturing. Insulation and Interface Adapter (IA) voltage breakdown measurements often require specially designed power supplies and loading fixtures. It is very difficult to generalize about verification of the functionality of interface adapter active circuitry. IA active circuitry may serve to eliminate costly ATE modification, improve noise immunity, and capture very high

frequency low duty cycle signals or other signal conditioning functions. The IA fixtures range in spectrum from simple supports to complex optic benches and environmental simulators (such as acoustic or complex electromagnetic environments). This uniqueness implies a requirement for unique operational verification procedures beyond the envisioned scope of the UUT simulator.

The remaining two subtasks, verification of continuity from the ATE to a UUT testpoint and isolation between test points, appear to be addressable by the UUT simulator. The results of our survey indicate that miswiring and poor workmanship were the most common IA problems. The debug of the IA appears to be a profitable area to devote some effort.

Anticipated problem areas in adapting the UUT simulator to this task are as follows:

- The variety of physical configurations used for present ATE/UUT IAs.

  Configuration differences range from different sized and interlocking connectors to conceptual differences such as octopus types of cables.

- The validity of the IA description to the UUT simulator.

  Although this problem may appear trite, it is believed that many adapter errors are due to the menial nature of the tasks of describing and building IAs.

Because the results of the survey indicate that validation puts heavy demands on ATE station time, it would appear wise to remove the IA verification task from the ATE if possible. Assuming, then, that an ATE interface is common to many UUTs, the problem of the variety of physical configurations can be addressed by building a UUT simulator to ATE interface adapter which then becomes common to all UUTs tested on that ATE (assuming that the UUT simulator consists of a set of hardware and software separate from the ATE).

SECTION 6

CONCLUSIONS

The following conclusions summarize our findings during the conduct of the UUT Simulator Feasibility Investigation Study program:

- The feasibility of simulators for classes of UUTs in the digital, analog, and hybrid area is clear; their cost-effectiveness in the support of V&V is still to be validated.

  There is an abundance of all levels of simulation techniques which are capable of being used for UUT simulations. The issues are related more to cost-effectiveness than feasibility. The computer systems technology which is available for realizing these simulations is becoming less expensive, implying a greater potential for UUT simulator practicality.

  There may be some UUTs which are not feasible to simulate fully; this fact has been recognized in the study and acknowledged through statements that the real UUT cannot be replaced fully with a UUT simulator.

- To be most effective, the UUT simulator development must start early in the development cycle of an LRU, and the simulator must be based on the LRU design or schematics.

The initial emphasis in our study was oriented toward minimizing test program development costs and V&V costs. Although this goal in itself is a valid one, direction from the Air Force indicated that the UUT simulator should be oriented more toward ensuring that the V&V process provides better checks on the TRD in addition to the test programs. The rationale is that if higher quality test programs going to the field result from the improved V&V, higher effective payoff will be realized by the Air Force. The validation of such a payoff is not supportable now with available data and would require monitoring over a number of years to be quantifiable.

- The UUT simulator should be used as an orthogonal check on the validity and completeness of the test programs. Assessment of the degree of completeness of fault detection and isolation of the test programs to minimize LRU down-time is a key payoff factor for the Air Force. This orthogonal check is realized if the simulator is developed independent of the team developing the test programs.

  Another check is afforded if the UUT simulator development and independent V&V teams view the UUT from a completely different perspective than the test program developers. When these two views are completely reconciled by having the test programs exercising the UUT simulator, a high degree of confidence in the test programs can be achieved.

- The UUT simulator should be based on multiple levels of simulation and on either a software-only or instrument bus interface with the ATE system on which the test programs are executed. Solutions based on analog signal interfaces are not recommended.

  It was determined that the analog signal interface solutions were not only costly to develop and use but also less desirable from a utility point of view. Such an interface may introduce extraneous problems into the V&V process, which may detract from the usefulness of the simulator. Comments resulting from the survey that was conducted also indicated low credibility in the cost-effectiveness of such solutions.

  From the available data, it was not possible to determine whether the software-only solution (i.e., the simulator running in the same computer as the test program interpreter) was more cost-effective than the solutions using an instrument bus interface to the ATE system. The choice must depend on the relative degree of standardization on the test programming language (ATLAS) and the degree of standardization on the bus interface used in the ATE system.

  The multi-level simulation recommendation stems from the fact that different levels of simulation can be used in an optimum manner, resulting in the most cost-effective usage in the long run. For specific uses, specific levels of simulation may be more cost-effective.

- In using the UUT simulator as recommended above, the TPS development cost may actually increase, while decreasing the maintenance life-cycle costs.

  The first reason for this potential increase is that the UUT simulator must also be developed. The second reason is that the orthogonal checks provided by the UUT simulator will find more bugs during the TPS development cycle, which may otherwise have slipped through acceptance testing. Therefore the "sell-off" of the TPS may be delayed. But the resulting increase in test program quality should more than offset the development cost increase.

- A UUT simulator is not a total or the only solution in supporting V&V and in decreasing test program development and maintenance costs.

  The simulator is not a total solution because the recommended forms of simulation leave some, even though ultimately small, part of the validation uncovered. For those checks the real UUT is still needed.

  There are other technologies, such as automatic test program generation, which if satisfactorily realized, can be more cost-effective in reducing test program development and maintenance costs than the use of a UUT simulator. For example, in those areas where digital ATPG applies, a UUT simulator is not needed, except, perhaps, to validate that the ATPG generates correct results.

A UUT simulator is most effective where ATPG does not apply, because circuit complexity is higher than that handled by the ATPG or those areas where ATPG has not yet been developed, or it is not cost-effective.

- Test program developers and validators must be convinced of the merits of a UUT simulator. A moderate amount of skepticism over the cost-effectiveness of a UUT simulator was encountered while conducting the survey. A demonstration program addressing further issues and supporting the validation of the UUT simulator concept is necessary before large scale usage of UUT simulators can be started.

- Additional UUT simulator cost-effectiveness issues must be resolved. At this point the nature of the ultimate recommendation on the use of a UUT simulator in the MATE context is not known; partly because the outputs of the MATE Program are not yet known. It appears that any one of the following is still a possible outcome of the UUT simulator demonstration program:
  - —The recommendation to develop a generic UUT simulator system, analogous to an ATE system, which is modular and can be tailored to quickly and cost-effectively realize specific UUT simulators.
  - —The recommendation to develop a family of less generic but lower cost UUT simulator systems that would be oriented to specific classes of UUTs.
  - —The recommendation to not construct any generic system, but rather to provide guidelines and perhaps software modules to support developing unique UUT simulators based on specific weapon program requirements.

REFERENCES

1. Accampo, P. W. "Automatic Resource Allocation: A Means for Reducing Software Support Costs of ATLAS Procedures." Autotestcon 78, 28-30 November 1978, pp. 361-364.

2. Akers, S. B. "Partitioning for Testability." Proceedings of the ATCAW, * April 1978, pp. 137-139.

3. Atlas, J., & Carlin, R. "Dynamic Buffering Expands Digital Test Capability." Autotestcon 76, 10-12 November 1976.

4. Balekdjain, K. G. "Hardware-Software Trade-offs for Analog Automatic Test Instruments." Autotestcon 76, 10-12 November 1976, pp. 61-64.

5. Barbacci, M. R. "Instruction Set Processor Specifications for Simulation, Evaluation, and Synthesis." Proceedings of the 16th Design Automation Conference, June 1979, pp. 64-72.

6. Bastian, J. D. "Nonlinear Device Modeling." Proceedings of the ATCAW, April 1978, pp. 84-85.

7. Bastian, J. D., Hochwald, W., & Suziki, F. T. "Figure of Merit for Bit Testability for VLSI. "Autotestcon 79, 19-21 September 1979, pp. 90-95.

8. Batchelder, O. R., & Sumner, G. C. "F-16 Test Program Verification and Validation: A GD/FW Point of View." Autotestcon 79, Supplementary paper.

9. Bedrosian, D. D., & Lee, J. H. "An Application of Fuzzy Measure for Analog Fault Isolation." Autotestcon 79, 19-21 September 1979, pp. 276-280.

10. Beiman, H. "The Test Equipment Effectiveness Model (TEEM)." Proceedings of the ATCAW, April 1978, pp. 511-512.

*Automatic Test Conference and Workshop

REFERENCES (continued)

11. Berman, R. W. NOPAL Processor: Intra-Test Sequence. (AD A053 315), Moore School of Electrical Engineering, Philadelphia, Pennsylvania, January 1978.

12. Bhujade, M. R., & Deshpande, S. R. "Microprocessor Testing Using a Minimal Instruction Cover." Autotestcon 79, 19-21 September 1979, pp. 182-186.

13. Brueuer, M. A., & Friedman, A. D. "Test/80: A Proposal for an Advanced Automatic Test Generation System." Autotestcon 79, 19-21 September 1979, pp. 305-312.

14. Brocchi, R. "NAEC Lessons Learned." Proceedings of the ATCAW, April 1978, pp. 452-455.

15. Broutt, D. "Test System Readiness Testing." Proceedings of the ATCAW, April 1978, p. 43.

16. Brown, P. S. "Resource Flow Analysis in an ATLAS Compiler." Autotestcon 79, 19-21 September 1979, pp. 121-125.

17. Busching, H. L., & Mixer, C. R. "Commercial Instruments for MATE Meeting the USAF Need for Mission Readiness Affordable by Design." Autotestcon 79, Supplementary paper.

18. Bush, T. S. "Local Memory for A High-Speed Digital Test System." Autotestcon 76, 10-12 November 1976, pp. 193-196.

19. Byrd, G. R. "Interface Effects on Test Program Sets." Proceedings of the ATCAW, April 1978, pp. 281-284.

20. Carter, W. C., Joyner, W. H., & Brand, D. "Symbolic Simulation for Correct Machine Design." Proceedings of the 16th Design Automation Conference, June 1979, pp. 280-286.

21. Case, G. R. Analysis of Actual Fault Mechanisms in CMOS Logic Gates. Unpublished paper.

REFERENCES (continued)

22. Chandler, W. "An Approach to Verification and Validation of Test Program Sets." Proceedings of the ATCAW, April 1978, pp. 456-457.

23. Chang, Y. K. Automatic Test Program Generation. (AD A054 910), Moore School of Electrical Engineering, Philadelphia, Pennsylvania, March 1978.

24. Chien, R. T. "On the Use of Deductive Models for Automatic Generation of Test Program for Analog Circuits." Proceedings of the ATCAW, April 1978, pp. 100-105.

25. Chien, R. T., & Ho, W. P. -C. "Strategy for Automatic Testing by Circuit Understanding." Autotestcon 79, 19-21 September 1979, pp. 254-260.

26. Chizmadia, R. "'Design To' Criteria for ATE Compatible Systems." (Abstract), Autotestcon 76, 10-12 November 1976, p. 111.

27. Chouinard, R. J. "Test Requirements Document (TRD)." Proceedings of the ATCAW, April 1978, pp. 361-362.

28. Clark, N. M., et al. "Indicators of ATE Test Program Quality." Autotestcon 77, 2-4 November 1977, pp. 261-269.

29. Clary, J. B., & Smith, F. M. "Verification of Built-in-Test Performance in Modular Digital Systems Using Instruction Set Processor (ISP) Language Descriptions." Autotestcon 79, 19-21 September 1979, pp. 96-101.

30. Colebank, J. M., et al. "Management of Test Program Development for S-3A." Autotestcon 77, 2-4 November 1977, pp. 246-253.

31. Colgan, J. "Cost Saving Approach to Automatic Test Equipment for F-18L Program." Autotestcon 79, 19-21 September 1979, pp. 78-80.

32. Cortner, J. M. "A User's View of the Next Step in Test Generation Software." Autotestcon 79, 19-21 September 1979, pp. 300-304.

REFERENCES (continued)

33. Cortner, J. M. "Testability Doesn't Cost--It Saves!" Autotestcon 79, 19-21 September 1979, pp. 281-285.

34. Crowley, D. F. "Design for Testability of PC Boards Containing Microprocessors." Proceedings of the ATCAW, April 1978, pp. 242-245.

35. Day, D. B. "Digital Automatic Test Program Generation Selection." Autotestcon 79, 19-21 September 1979, pp. 295-299.

36. Degerman, D. A., & Gustafson, A. R. "Automatic Testing of Microwave Avionics." Autotestcon 76, 10-12 November 1976, pp. 101-110.

37. Dejka, W. J. "A Review of Measures of Testability for Analog Systems." Autotestcon 77, 2-4 November 1977, pp. 279-284.

38. Delger, K. R. "Total Life-Cycle Cost Model." Proceedings of the ATCAW, April 1978, pp. 344-345.

39. Dooley, J. M. "VAST Compatibility Program (VCOMP): A UUT Requirements Analysis Tool." Autotestcon 78, 28-30 November 1978, pp. 389-394.

40. Drown, G. A. "A Cost-Effective Approach to ATE." Autotestcon 79, 19-21 September 1979, pp. 229-232.

41. Dunning, B. B. "Self-Learning Data Base for Automated Fault Localization." Autotestcon 79, 19-21 September 1979, pp. 155-157.

42. Durgavich, J. J., et al. "Consideration for, and Impact of, the ATE/UTT Interface." IEEE Trans. 1978, Instrum. and Meas., Vol. IM27, No. 2, June 1978, pp. 132-136.

43. Durgavich, J. J., & Koester, F. "Cost Benefit Analysis." Autotestcon 79, 19-21 September 1979, pp. 212-216.

44. Ellis, G. "Microcomputer Controlled Precision Pneumatic Pressure Generator." Autotestcon 76, 10-12 November 1976, pp. 188-192.

45. Ellis, M. T. "OPAL/ATLAS: A Machinability Analysis." Proceedings of the ATCAW, April 1978, p. 434.

46. Elwell, D. F. "Honeywell ATE: Software Features Which Improve ATE Economics." Proceedings of the ATCAW, April 1978, pp. 515-517.

47. El-Zig, Y. "Testing of MOS Combinational Networks: A Procedure for Efficient Fault Simulation and Test Generation." Proceedings of the 16th Design Automation Conference, June 1979, pp. 162-170.

48. Emero, R. F., & Holmquest, C. A. "Patriot Weapon System Testability and Maintainability Philosophy." (Abstract), Proceedings of the ATCAW, April 1978, pp. 110-111.

49. Emilo, G., et al. "New Automatic Test Equipment Emphasizes User Interface." Autotestcon 77, 2-4 November 1977, pp. 191-198.

50. Fay, J. E., & Eiranova, A. "Can Software Portability Alleviate Proliferation Headaches?" Autotestcon 79, 19-21 September 1979, pp. 117-120.

51. Ferrel, J. "Logical Planning for ATE Software Requirements." Proceedings of the ATCAW, April 1978, pp. 483-491.

52. Fleming, W. R. "Requirements Communication." Proceedings of the ATCAW, April 1978, pp. 31-32.

53. Forster, E., & Eichna, O. "An Advanced RF Test System." (Abstract), Autotestcon 76, 10-12 November 1976, p. 46.

54. Fulgham, D. P. "Automatic Self-Test of A Microprocessor System." Autotestcon 76, 10-12 November 1976, pp. 47-52.

55. Fuller, O. G., & Haverbach, M. "ATE: A CRT Display Evaluation System." Autotestcon 79, 19-21 September 1979, pp. 197-205.

56. Garmer, D. M., & Fish, J. W. "Automatic Test Equipment Software Testing." Autotestcon 78, 18-30 November 1978, pp. 71-75.

REFERENCES (continued)

57. Gauthier, R. L., & de Mare, G. M. "A Digital Test Language for ATLAS." Autotestcon 79, 19-21 September 1979, pp. 46-50.

58. Giambiasi, N., Miara, A., & Muriach, D. "SILOG: A Practical Tool for Large Digital Network Simulations." Proceedings of the 16th Design Automation Conference, June 1979, pp. 263-271.

59. Giordano, P. J. "Advanced ATE Technology." Proceedings of the ATCAW, April 1978, pp. 249-250.

60. Goodenough, J. B., & Braun, C. L. Simulation Higher Order Language Requirements Study. (AD A058 994), Wright-Patterson Air Force Base, Dayton, Ohio, August 1978.

61. Gooding, M. J. "Current Approaches to Maintenance of DoD ATE Software: Issues and Challenges." Proceedings of the ATCAW, April 1978, pp. 41-42.

62. Gossman, A. "Improvements in MIL-STD-1388 Logistic Support Analysis." Proceedings of the ATCAW, April 1978, pp. 358-360.

63. Gracia, J. A., et al. Multiplex Simulator Design Study. (AD A037 226), Harris Corp., Melbourne, Florida, January 1977.

64. Granieri, M. N., et al. "An Approach to ATE/UUT Functional Matching." Autotestcon 78, 28-30 November 1978, pp. 351-360.

65. Granieri, M. N., & Schmitt, W. J. "A Comparative Analysis of Current "Portable" Digital ATE System Architectures." Autotestcon 79, 19-21 September 1979, pp. 10-17.

66. Graube, M. "The IEEE-488 Interface Standard: A Good Beginning." Proceedings of the ATCAW, April 1978, pp. 267-268.

67. Greenhow, W. A. "Cost-Drivers Affecting the Development of F-16 Depot ITA (Hardware) and Test Software." Autotestcon 79, 19-21 September 1979, pp. 136-140.

68. Greenspan, A. M. "ATE Hardware Standardization." Proceedings of the ATCAW, April 1978, pp. 251-253.

69. Hand, P. J. "Development, Production, and In-Service with Truly Reconfigurable ATE." Autotestcon 79, 19-21 September 1979, pp. 152-154.

70. Hanes, L. F., et al. "Human Aspects of ATE." Proceedings of the ATCAW, April 1978, pp. 272-274.

71. Hanson, P. A Test Strategy for Microprocessor-Based LSI Boards. Unpublished report.

72. Harrison, M. M. "Compiler Development: An Approach to Reducing UUT Software Maintenance Costs." Autotestcon 79, 19-21 September 1979, pp. 126-127.

73. Hart, D. C., et al. "A Computer-Independent ARINC ATLAS Syntax Comparator." Autotestcon 76, 10-12 November 1976, pp. 10-18.

74. Hartman, R. "Paper No. 14: Definition of Analog Faults." Proceedings of the ATCAW, April 1978, p. 94.

75. Henckels, L. P., et al. "Evaluation Criteria for Test Program Generation Systems." Proceedings of the 1978 Semiconductor Test Conference, 31 October-2 November 1978, pp. 76-78.

76. Henckels, L. P., et al. Selection Guide for Digital Test Program Generation Systems. Henckels, Haas and Brown, Inc., Upper Saddle River, New Jersey.

77. Henckels, L. P., et al. Selection Guide for Digital Test Program Generation Systems (Appendices). Henckels, Haas and Brown, Inc., Upper Saddle River, New Jersey.

78. Hendrickson, D. A. "ATE Software Cost Drivers in Relation to RCA Price S Parameters." Proceedings of the ATCAW, April 1978, pp. 34-39.

79. Herbert, D. "Compile Time Costs Money." Autotestcon 79, 19-21 September 1979, pp. 129-135.

80. Hill, D., & vanCleemput, W. M. "SABLE: A Tool for Generating Structured Multi-Level Simulations." Proceedings of the 16th Design Automation Conference, June 1979, pp. 272-279.

81. Hosking, M. B. "A New UUT/Test Station Interface." Autotestcon 78, 28-30 November 1978, pp. 346-350.

82. Howley, P. P., et al. "Software Verification for Large-Scale ATE Systems." Autotestcon 78, 28-30 November 1978, pp. 76-83.

83. Jackson, P. C. "Evolutionary Trends in Pin Electronics." Autotestcon 79, 19-21 September 1979, pp. 18-22.

84. Jacoby, D. O., & Ellis, W. L. "Advanced Electronic Warfare Test Set: Realistic or Futuristic." Autotestcon 79, 19-21 September 1979, pp. 179-181.

85. Jeschke, A. W., Kelly, R. B., et al. "Let Your Computer Write Your Test Program." Autotestcon 79, 19-21 September 1979, pp. 36-39.

86. Johnson, E. E. "The TRD Document: A Key To Improved Electronic Equipment Support." Proceedings of the ATCAW, April 1978, pp. 363-365.

87. Johnson, W. A., Crowley, J. J., & Ray, J. D. Mixed-Level Simulation From a Hierarchical CHDL. SIGDA Newsletter, Volume 10, No. 1, January 1980.

88. Joseph, S. "'Intelligent' System Switching." Autotestcon 76, 10-12 November 1976, pp. 185-187.

89. Kano, H. "Test Pattern Generation for Logic Networks by Real Number Logic Simulation." Autotestcon 79, 19-21 September 1979, pp. 168-178.

90. Kaplan, G. "Computer-Aided Design." IEEE Spectrum, Vol. 12, No. 10, October 1975, pp. 40-47.

91. Karsch, R. O., Hegner, H. R., & McWhirter, W. R., Jr. "The U.S. Navy's Shipboard Machinery Performance Monitoring System Development Program." Autotestcon 79, 19-21 September 1979, pp. 320-323.

92. Katz, E. H. "Universal Automatic RF Test Station." Autotestcon 76, 10-12 November 1976, pp. 180-184.

93. Keiner, W. L., et al. "Testability Measures." Autotestcon 77, 2-4 November 1977, pp. 49-55.

94. Keiner, W. L. "Testability Design Process." Proceedings of the ATCAW, April 1978, pp. 121-123.

95. Kenney, J. W. "Support Systems for Advanced Military Electronics." Autotestcon 77, 2-4 November 1977, pp. 64-71.

96. Kjelkerud, E., & Thessen, O. T. "Methods of Modeling Digital Devices for Logic Simulation." Proceedings of the 16th Design Automation Conference, June 1979, pp. 235-241.

97. Kole, R. S. "A Proposed System Architecture for Depot ATE." Autotestcon 77, 2-4 November 1977, pp. 99-106.

98. Kole, R. S. "Test Program Set Quality." Proceedings of the ATCAW, April 1978, pp. 444-446.

99. Kovijanic, P. G. "Computer-Aided Testability Analysis." Autotestcon 79, 19-21 September, pp. 292-294.

100. Krafcik, S., & Markee, H. "Analysis and Definition of Test Requirements." Autotestcon 79, 19-21 September 1979, pp. 345-348.

101. Kulig, D. M., Plaisted, R. C., & Plunkett, T. J. "Equipment Specification for ATE Testability." Autotestcon 79, 19-21 September 1979, pp. 81-84.

102. Ledford, K. H., & Nickerson, (CPT) T. J. "F-16 Avionics Intermediate Shop (AIS) User Involvement During Development." Autotestcon 79, 19-21 September 1979, pp. 330-333.

103. Liu, R. W. "Fault Diagnosis of Large-Scale Analog Systems: A Tearing Method Approach." Proceedings of the ATCAW, April 1978, pp. 91-94.

104. Long, (CPT) F. D., Jr., & Nodinge, (CPT) W. A., Jr. "Automatic Testing and Weapon System Acquisition Process." Autotestcon 79, 19-21 September 1979, pp. 206-211.

105. Loveman, D. B. "An ATE Language Processing System." Autotestcon 76, 10-12 November 1976, pp. 1-9.

106. Loveman, D. B. "Language Design for Automatic Test Equipment." Autotestcon 77, 2-4 November 1977, pp. 78-88.

107. Ludvigson, M. T. "Built-in Test in MIL-STD-1553 Systems." Autotestcon 79, 19-21 September 1979, pp. 102-106.

108. Manigian, G. "Interactive Fault Trace: A Simulator Aid For Test Program Development." Autotestcon 79, 19-21 September 1979, pp. 158-162.

109. Maxwell, R. L., & Miller, D. F. "Test System Requirements for 767 Aircraft Electrical Components." Autotestcon 79, 19-21 September 1979, pp. 349-358.

110. McGarvey, R. L., Tucker, R. W., & Weger, J. R. "A Practical ATLAS Implementation." Autotestcon 79, 19-21 September 1979, pp. 40-45.

111. McGrath, T. Data Item Description for Acquisition of Detailed Performance Characteristics and Signal Tracing Diagrams for Electronics. (AD A026 953), Naval Air Engineering Center, Lakehurst, N.J., April 1976.

112. Milkie, R. W. "Commercial Test Software Development Practices For Military Applications." Autotestcon 77, 2-4 November 1977, pp. 254-260.

113. Mills, M. T. "A Resource Allocation Analyzer for Source Programs Written in ATLAS." Autotestcon 78, 28-30 November 1978, pp. 365-369.

114. Modi, M., Bauer, J., & Epstein, R. "Navy Program for Development of an Analog Test Program Generation System." Autotestcon 79, 19-21 September 1979, pp. 250-253.

115. Modrow, M. B. "Language Driven ATE Architecture Can Save Money." Autotestcon 79, 19-21 September 1979, pp. 27-29.

116. Moebus, F. L. "Universal Pin Concept." Proceedings of the ATCAW, April 1978, pp. 275-278.

117. Moebus, F. L. "A Non-IEEE Bus View of the Computer Interface." Proceedings of the ATCAW, April 1978, pp. 279-280.

118. Moebus, F. L. "Distributed Instrumentation and Distributed Processing: Their Architectural Implications." Autotestcon 79, 19-21 September 1979, pp. 1-9.

119. Montenes, F., & Jain, G. "Systematic Analog Network Testing Approach: SANTA." Autotestcon 76, 10-12 November 1976, pp. 25-31.

120. Morrison, R. A. "Avionics-Universal Connector-Tester." Autotestcon 76, 10-12 November 1976, pp. 65-69.

121. Morton, (CPT) R. W., et al. A Methodology for Analysis of Alternatives in the Acquisition and Support of Automatic Test Equipment Software. (AD A004 163), Air Force Institute of Technology, Wright Patterson AFB, Ohio, June 1978.

122. Mueller, R. S. "Current Techniques to Reduce Software Development Time and Cost." Autotestcon 76, 10-12 November 1976, pp. 32-37.

123. Nagel, L. W. SPICE 2: A Computer Program to Simulate Semiconductor Circuits. Electronics Research Laboratory, College of Engineering, University of California, Berkeley California, 9 May 1975.

124. Navabi, Z., & Hill, F. J. "Efficient Simulation of AHPL." Proceedings of the 16th Design Automation Conference, June 1979, pp. 255-260.

125. Neuman, G. W. "The Navy Program in Automatic Testing." Autotestcon 77, 2-4 November 1977, pp. 9-14.

126. Nielsen, R., & Andreano, R. "'Smart' Buffer Resolves Classic ATE Problems." Autotestcon 76, 10-12 November 1976, pp. 38-45.

127. Novak, F. V., et al. ATE Software Life-Cycle Cost Simulation Model Validation. (AD A059 182), Air Force Institute of Technology, Wright Patterson AFB, Ohio, June 1978.

128. Nugent, J., & Fetridge, P. V. "On Defining and Obtaining Documentation Relative to ATE." Autotestcon 79, 19-21 September 1979, pp. 147-149.

129. Nuspl, S. An Incremental Charge Method for the Analysis of Nonlinear Circuits. Digital Computer Laboratory, University of Illinois, Urbana, Illinois, 6 August 1964.

130. O'Connor, (MAJ) P. D. "F-16 LRU Test Programs: A Systems Approach." Autotestcon 77, 2-4 November 1977, pp. 270-278.

131. Owens, P. R. "The Challenge of New Technology for Avionics Testing." Proceedings of the ATCAW, April 1978, pp. 201-203.

132. Pau, L. F., Wafae, B., & Bousquet, A. "Fault Detection Capability Implications for Flow Control in Data Communication Networks." Autotestcon 79, 19-21 September 1979, pp. 359-364.

133. Paulsen, W. E. "Common GSE Under Microprocessor Control." Autotestcon 76, 10-12 November 1976, pp. 54-60.

134. Pearson, T. E., et al. Analysis of Software Simulation in Computer-Based Electronic Equipment Maintenance Trainer. (AD A060 583), Training Analysis and Evaluation Group, Orlando, Florida, September 1978.

135. Pedersen, G. "Reducing ATE Test Software Development Costs." Autotestcon 79, Supplementary paper.

136. Plice, W. A. "Digital Filters as Analog Circuit Models." Proceedings of the ATCAW, April 1978, pp. 86-87.

137. Plice, W. A. "I-B ATPG Recommendations." Proceedings of the ATCAW, April 1978, pp. 532-537.

138. Plice, W. A. Overview of Current Automatic Analog Test Design. Paper presented at Cherry Hill Test Conference, 23-25 October, 1979.

139. Pomeroy, B. A. "Lasar: Quality Assurance for Digital Field Testing." Autotestcon 79, 19-21 September 1979, pp. 57-60.

140. Prywes, N. S. "ATLAS/OPAL Comparison and Evaluation Study." Proceedings of the ATCAW, April 1978, pp. 432-433.

141. Purks, S. R. "Experiences with ATE Providing Testability of Microprocessor Boards." IEEE Trans. 1978 Instrum. and Meas. Vol. IM27, No. 2, June 1979, pp. 178-181.

142. Raymond, L., & Leo, R. "CAPS: An Enhanced Interactive Simulation." Autotestcon 79, Supplementary Paper.

143. Reifer, D. J., & Trattner, S. "A Glossary of Software Tools and Techniques." IEEE Computer, Vol. 10, No. 7, July 1977, pp. 52-60.

144. Richter, E. W. "Designing Microwave ATE to Meet UUT Requirements." Autotestcon 76, 10-12 November 1976, pp. 87-100.

145. Rickers, H. C. Microcircuit Device Reliability: Memory/LSI Data." IIT Research Institute, January 1978.

146. Ring, S. J. "Automatic Testing Via a Distributed Intelligence Processing System." Autotestcon 77, 2-4 November 1977, pp. 89-98.

147. Ring, S. J. "A Distributed Intelligence Automatic Test System for PATRIOT." IEEE Trans. 1977, Aerosp. and Electron Systems, Vol. AES-13, No. 3, May 1977, pp. 264-272.

148. RTL CAD: User Reference Manual. CAT Administration, Department 707, Aerospace, Florida, 20 April 1973.

149. Saeks, R. "An Experiment in Fault Prediction-II." (Abstract), Autotestcon 76, 10-12 November 1976, p. 53.

150. Salter, M. W. "In-Circuit vs. Functional PC-Board Testing Trade-offs." Unpublished paper.

151. Salzmann, C. H. "Automatic Test Sequence Generation." Autotestcon 76, 10-12 November 1976, pp. 112-116.

152. Santo, J. C. "PATEC: An Air Force Approach to ATE Calibration." Autotestcon 79, 19-21 September 1979, pp. 242-246.

153. Santoni, A. "Automatic Test Generators Crank Out Programs Quickly." EDN, Volume 25, No. 4, 20 February 1980, pp. 38-41.

154. Schlosser, S. M. "The Third Generation Concepts: A Perspective." Autotestcon 76, 10-12 November 1976, pp. 151-157.

155. Schmitt, W. J., & Paffenroth, G. G. "Selecting an ATE on a Limited Parametric Basis." Autotestcon 79, 19-21 September 1979, pp. 217-228.

156. Schnable, G. L., Gallacs, L. J., & Pujol, H. L. "Reliability of CMOS Integrated Circuits." IEEE Computer, Vol. 11, No. 10, October 1978, pp. 6-17.

157. Schneider, R. D. "Standardization of Design Guides, Plans, Terms, and Definitions." Proceedings of the ATCAW, April 1978, pp. 126-128.

158. Scholten, R. W. "Configuration Management For Test Program Sets." Proceedings of the ATCAW, April 1978, pp. 447-451.

159. Schreiber, H. H. "A State Space Approach to Analog Automatic Test Generation." Proceedings of the IEEE NAECON 1977, May 1977, pp. 797-804.

160. Schreiber, H. H. "Fault Dictionary Construction Using the Complementary Signal." Proceedings of the IEEE NAECON 1978, 16-18 May 1978, pp. 1160-1168.

161. Schreiber, H. H. "An Overview of Analog Fault Isolation Techniques." Proceedings of the ATCAW, April 1978, pp. 77-80.

162. Schreiber, H. H. "A Review of Analog Test Generation." Autotestcon 78, 28-30 November 1978, pp. 1-8.

163. Schwedner, F. A. "In-Circuit Testing: Another Approach." Autotestcon 79, 19-21 September 1979, pp. 53-56.

164. Scott, E. "MIL-STD-1388 In Automatic Test Equipment Acquisition." Proceedings of the ATCAW, April 1978, pp. 356-357.

165. Scully, J. K. "Acquisition of Test Compatible Avionics: An Updated Approach." Autotestcon 76, 10-12 November 1976, pp. 143-149.

166. Scully, J. K. "Blue Sky R&D." Proceedings of the ATCAW, April 1978, pp. 259-261.

167. Seening, Y. "Modeling Technique for Unconventional Switching Networks in Automatic Test Generation." Autotestcon 79, 19-21 September 1979, pp. 163-167.

168. Selby, C. D. "Edge Testing an Inexpensive Alternative." Autotestcon 76, 10-12 October 1976, pp. 124-130.

169. Sherwood, W. "A Hybrid Scheduling Technique for Hierarchical Logic Simulators." Proceedings of the 16th Design Automation Conference, June 1979, pp. 249-254.

170. Shumovich, J. V., & Tucker, R. E. "Advanced Concepts for Component Failure Isolation." Autotestcon 76, 10-12 November 1976, pp. 19-24.

171. Sick, N. R., Tucker, R. W., Robinson, R. L., & Rytter, L. J. "The Impact of Complex Digital UUTs on the Use of Computer Programs to Automatically Generate Test Programs." Autotestcon 79, 19-21 September 1979, pp. 313-316.

172. Sides, J. B. "Air Force Automatic Test Equipment (ATE) Life-Cycle Technical and Logistics Support Considerations." Autotestcon 79, 19-21 September 1979, pp. 141-146.

173. Stroud, E. A. "Test Program Set Content." Proceedings of the ATCAW, April 1978, pp. 443.

174. Tehan, J. "Rehostable ATE System Software." Autotestcon 79, 19-21 September 1979, pp. 30-35.

175. Thomas, J. J. "Gate-Level Modeling for Digital ATPG." Proceedings of the ATCAW, April 1978, pp. 62-64.

176. Thompson, J. R., et al. "Digital Logic Simulation Aids Utilized at McDonnell Aircraft Company (MCAIR)." Autotestcon 78, 28-30 November 1978, pp. 31-38.

177. Timoc, C. C., & Hess, L. M. Fault Simulation: An Implementation into Hardware. Jet Propulsion Laboratory, Pasadena, California.

178. Tinaztepe, C., et al. "Automatic Test Program Generation." Autotestcon 77, 2-4 November 1977, pp. 240-245.

179. To, K., et al. "Automatic Test Systems (Circuit Testing)." IEEE Spectrum, Vol. II, No. 9, September 1974, pp. 44-52.

180. Tobias, R. W., & Heslin, R. F. "Test Program Set Development Process Used for P3 ORION Avionics Support." Autotestcon 79, 19-21 September 1979.

181. Toscano, P. M. "Cost Estimating Relationships for ATE Test Program Set Development." Proceedings of the ATCAW, April 1978, pp. 366-370.

182. Tucker, R. E. "Computer-Aided Design Application to S-3A Analog Test Programs." Autotestcon 76, 10-12 November 1976, pp. 172-179.

183. vanCleemput, W. M. "An Hierarchical Language for the Structural Description of Digital Systems." Proceedings of the 14th Design Automation Conference, 19 June 1977, pp. 377-385.

184. Van Hemel, P. E. "Operator and Technician Tasks for the Heads-Up Display Test Set and Versatile Avionics Shop Test (VAST)." Proceedings of the ATCAW, April 1978, pp. 392-394.

185. Van Hemel, P. E. "Operator and Technician Tasks for the Heads-Up Display Test Set and Versatile Avionics Shop Test (VAST)." Proceedings of the ATCAW, April 1978, pp. 411-414.

186. Vartanian, M. M. "A Program for Fault Analysis of Analog Networks." Proceedings of the ATCAW, April 1978, pp. 97-99.

187. Wakefield, W. R., & Kuehn, H. S. "The Future Role of Systems Engineering." Proceedings of the ATCAW, April 1978, pp. 333-336.

188. Walker, (CPT) H. M. "Joint Service Planning and Coordination." Autotestcon 77, 2-4 November 1977, pp. 15-18.

189. Wang, F. L. "Digital System Modeling for ATG." Proceedings of the ATCAW, April 1978, pp. 58-61.

190. Wang, F. L. "On Feasibility of ATG for Hybrid Systems." Autotestcon 79, 19-21 September 1979, pp. 268-275.

191. Weber, M. B. "Testability: The Key to Economical and Operationally Effective Avionic Test Software." Autotestcon 78, 28-30 November 1978, pp. 300-304.

192. West, R. P., & McCoy , W. L. "A Real-Time Verification and Evaluation System." Autotestcon 79, 19-21 September 1979, pp. 334-341.

193. Whealan, J. L. "Interactive Testing of Digital Subsystems (LRUs)." Autotestcon 76, 10-12 November 1976, pp. 135-142.

194. Whisnant, R. A., et al. AFAL Simulation Facility/Capability Manual." (AD A055 591), Research Triangle Institute Research Triangle Park, North Carolina, June 1977.

195. White, (CPT) N. E. "F-16 Independent Assessment: An Air Force Viewpoint." Autotestcon 79, 19-21 September 1979, pp. 342-344.

196. Wilkinson, (LTC) K. D. "Summary of the Modular ATE (MATE) Program." Autotestcon 77, 2-4 November 1977, pp. 7-8.

197. Willging, A. "A Special Purpose Data Bus for Unique and Custom Designed Electronics." Autotescon 79, 19-21 September 1979, pp. 23-26.

198. Wilson, Q. F., & Day, D. B. "Practical Automatic Test Program Generation Constraints." Proceedings of the ATCAW, April 1978, pp. 73-76.

199. Wood, C. T. "The Quantitative Measure of Testability." Autotestcon 79, 19-21 September 1979, pp. 286-291.

200. Zimmerman, A. "Trends in IEEE-488 Compatible Products." Proceedings of the ATCAW, April 1978, pp. 269-271.

201. Zingg, D. J. "Life-Cycle Costs of Test Program Sets." Proceedings of the ATCAW, April 1978, pp. 458-460.

APPENDIX A

REFERENCE CONTENT CLASSIFICATION

REF	MAIN AUTHOR	DOES PAPER DISCUSS													RELEVANCE CLASSIFICATION A = HIGH D = LOW BLANK = NONE
		UUT REFERENCE	ID REFERENCE	TRD REFERENCE	ANALOG UUT	DIGITAL UUT	TEST PROG DEV. COSTS	TPS DEV. DIFF'S.	UUT DIFF-ICULTIES	ID DIFF-ICULTIES	ATE DIFF-ICULTIES	PAYOFF INFORMATION	DEVELOPMENT TOOLS		COMMENT
1	Accomro	C	C								B		A		Auto. Resource Alloc.
2	Akers	D							B				D		Partitioning For Testability
6	Bastian				B			B					D		Nonlinear Device Modeling Discussion
10	Beiman		D										C		TEEM ATE Selection Prog. Not Relevant
14	Brocchi	C	C	C			D	C	C				D		NAEC Lessons Learned in TPS Dev.
19	Byrd	A	A					B		A					Interface Effects on TPS
22	Chandler	C						B			C				S-3A Vast Approach to TPS V&V
24	Chien	B			B			B					B		Deductive Models For AATPG
27	Chouinard		C	A				B							TRD: Army Problems and Proposed Solutions
28	Clark	C			B	B		D	D	D	B				TPS Quality Measure (First Find Ratio)
30	Colebank	B	B	B			A	A	C	B	C		C		Good Insight Into TPS Development
37	Dejka	C			A			B					A		Testability And Analog Circuit AATPG
38	Delger											D			Outline of Potential Life Cycle Cost Model
39	Dooley	C											B		ATE/UUT Comp Analysis (VAST)
42	Durgavich	B	A	C	B	B			C	A	C				Cautions Against General ATE/UUT ID

REF	MAIN AUTHOR	DOES PAPER DISCUSS													RELEVANCE CLASSIFICATION A = HIGH D = LOW BLANK = NONE
		UUT REFERENCE	ID REFERENCE	TRD REFERENCE	ANALOG UUT	DIGITAL UUT	TEST PROG DEV. COSTS	TPS DEV. DIFF'S.	UUT DIFF-ICULTIES	ID DIFF-ICULTIES	ATE DIFF-ICULTIES	PAYOFF INFORMATION	DEVELOPMENT TOOLS		COMMENT
45	Ellis	C	C								D				ATLAS/OPAL Machinability
46	Elwell	C			C	C		C			D		B		Honeywell ATE Features
48	Emero														Patriot Testability and Maintainability
49	Emilio	A	B		A	B		D	C	C	C		C		Grumman's CAT-A ATE
51	Ferrel	D	C	C			A	B		C	C	A	D		Has TPS Life Cycle Cost Examples
52	Fleming	C		B				C							Discusses Requirements Comm. Improve.
56	Garmer	D		C									D		Testing of ATE Software. Shallow
59	Giordano	D		C					D		D				Recommendations on Advanced ATE Tech.
61	Gooding							C			C		C		ATE Maintenance Issue
62	Gossman										B				Improvements in M1388 Not Relevant
64	Granieri	A	D	B	B	B							C		ATE/UUT Matching 1048 UUTs
66	Graube								C	B			B		IEEE-488 Bus In ATE
68	Greenspan									C			C		ATE Hardware Standardization in Services
70	Hanes	D								B					Human Aspects of ATE
74	Hartman	D			B										Brief Definition of Analog Faults

REF	MAIN AUTHOR	DOES PAPER DISCUSS													RELEVANCE CLASSIFICATION A = HIGH D = LOW BLANK = NONE
		UUT REFERENCE	ID REFERENCE	TPS REFERENCE	ANALOG UUT	DIGITAL UUT	TEST PROG DEV. COSTS	TPS DEV. DIFF'S.	UUT DIFF-ICULTIES	ID DIFF-ICULTIES	ATE DIFF-ICULTIES	PAYOFF INFORMATION	DEVELOPMENT TOOLS		COMMENT
75	Henckels	D				B		C					C		Criteria Used in Evaluating Dig. ATPG
78	Hendrickson	C										B			Lists ATA Software Cost Drivers.No Res.
81	Hosking	C	A							B	B				F-16 UUT/ATE Interface
82	Howley	B	D	B			D	C				C	C		Verification Method UUT Simulator Require.
86	Johnson	B		A				C	B				B		TRD Usage Benefits
93	Keiner	C	C					C	B				C		Testability Measures (NAVY)
94	Keiner	C							A						Testability Concepts and Measures
95	Kenney	D						C	D						Has Maintenance Trend Table
97	Kole	B	D	C				C					D		TPS Quality Attributes
98	Kole	B		D							B	B	C		ATE Architecture Alternatives
103	Liu	D			B				C				D		Tearing Method for FI in Analog Circuits
106	Loveman	B	C	C									A		ATE Language Issues
112	Milkie	B	C	B				C					D		TPS Development Phase Activities Documen.
113	Mills												B		Resource Allocation Analyzer
116	Moebus	B	A							B			B		Strong Position for Universal Pin Concept

REF	MAIN AUTHOR	DOES PAPER DISCUSS													RELEVANCE CLASSIFICATION A = HIGH D = LOW BLANK = NONE
		UUT REFERENCE	ID REFERENCE	TRD REFERENCE	ANALOG UUT	DIGITAL UUT	TEST PROG DEV. COSTS	TPS DEV. DIFF'S.	UUT DIFF-ICULTIES	ID DIFF-ICULTIES	ATE DIFF-ICULTIES	PAYOFF INFORMATION	DEVELOPMENT TOOLS		COMMENT
117	Moebus										B				Cautions Against 488 In Comp./ATE Interface
125	Neuman	D		D				C							NAVY ATE Program
130	O'Connor	C	C	A	D	D		B	D	C			C		F-16 TPS Development Approach
131	Owens	D							B						Where Technology Is Going
137	Plice	C			B	B		B	B				B		ATPG Recommendations
136	Plice	C			B								C		Digital Filters as Models for AATPG
140	Prywes							C					B		ATLAS/OPAL Comparison
141	Purks	D				C		C	C				D		Testability of Microprocessor Boards
146	Ring	A	D		B	B							C		Ratheon ATE Ratel Language
147	Ring	B	D		C	C							A		Ratheon ATE Ratel Language
157	Schneider								B						Testability Design Guides
158	Scholten	C	C	C				C					D		Config Management for TPS
160	Schreiber	B			B			C					A		Fault Dictionary for AATPG
162	Schreiber	C			B	D		B	C				C		AATPG Status and Approaches
161	Schreiber	D			B								B		AATPG Fault Isolation (similar to Ref 1 & 2)

REF	MAIN AUTHOR	DOES PAPER DISCUSS													RELEVANCE CLASSIFICATION A = HIGH D = LOW BLANK = NONE
		UUT REFERENCE	ID REFERENCE	TRD REFERENCE	ANALOG UUT	DIGITAL UUT	TEST PROG DEV. COSTS	TPS DEV. DIFF'S.	UUT DIFF-ICULTIES	ID DIFF-ICULTIES	ATE DIFF-ICULTIES	PAYOFF INFORMATION	DEVELOPMENT TOOLS		COMMENT
159	Schrieber	D			B								A		State Space Approach for AATPG
164	Scott										D				MIL STD 1388 Logistics Support Analysis
166	Scully	C									D		B		Blue Sky R & D Areas for ATE
173	Stroud	B	B										C		TPS Content, Not Much in Paper
175	Thomas	D				A			C		C		C		Strongly Against Func. ATG Dig. Sim.
176	Thompson	B	D			A		C	C				A		Dig.Log.Sim.for ATPG Need for Sim. Aids
178	Tinaztepe	B	D	D									A		Discusses NOPAL
179	To	B			C	C	A	B	D		B		B		Discusses Required Charac. of Good ATE
181	Toscano	B	C	D	D	D	C	B			D	B	C		TPS Dev. Cost Estimating
184	Van Hemel	C	C								C				Operator and Tech. Tasks not Relevant
185	Van Hemel	C	C								C				Similar to 139
186	Vartanian	A			B	B		C					B		Fault Analysis of Analog UUTs
187	Wakefield										D				Role of Systems Engineer
188	Walker	D	D	D											Joint Services Planning
189	Wang	B				A			C		C		B		Relevant Comments on Digital Modeling

REF	MAIN AUTHOR	DOES PAPER DISCUSS													RELEVANCE CLASSIFICATION A = HIGH D = LOW BLANK = NONE
		UUT REFERENCE	ID REFERENCE	TRD REFERENCE	ANALOG UUT	DIGITAL UUT	TEST PROG DEV. COSTS	TPS DEV. DIFF'S.	UUT DIFF-ICULTIES	ID DIFF-ICULTIES	ATE DIFF-ICULTIES	PAYOFF INFORMATION	DEVELOPMENT TOOLS		COMMENT
191	Weber	C	C	B	D	D	A	B		C	C	B	C		TPS Dev.Cost Factors Des. for Testability
196	Wilkinson	D	D	D				C					D		MATE Program (2 page summary)
198	Wilson	C			C	C		C	C				C		Discusses Practical ATPG Constraints
200	Zimmerman									C	C		B		Trends in IEEE-488 Products
201	Zingg	C	C	D				C	C			C	D		TPS Life Cycle Cost Factors

APPENDIX B

SPECIFIC EXAMPLES OF STIMULI AND RESPONSE CLASSIFICATIONS FOR TRDs

Example 1:

Stimulus - level 2; response - level 2
RF amp TP 67

- Part 1

  A. Stimulus

     1. Vary power level at 2JI for band 1 and band 2.

        a. Band 1 is (F2 + F5)/2.
        b. Band 2 is (F9 + F4)/2.

     2. Vary CW inputs from -17.5 dBm to -6.5 dBm in 1.0 dBm increments.

  B. Response

     1. Measure the RF power at each increment.

        a. Band 1, power is P1C min.
        b. Band 2, power is P2C min.

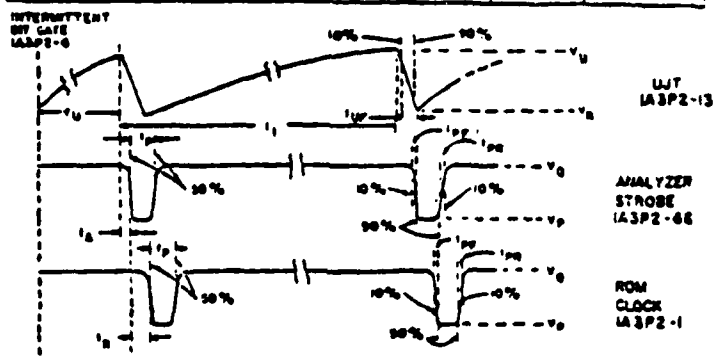
Example 2:

Stimulus - level 1; response - level 8
Fault locator, TP 32

A. Apply a 1 usec negative going pulse to connections 1A3P2-5 and 1A3P2-26.

B. The UTT, analyzer strobe, and ROM clock will have the following characteristics:

CHARACTERISTIC	UNITS	LOW LIMIT	NOMINAL	HIGH LIMIT
ROM CLOCK DELAY, t_q	μs	1.8	2.0	2.2
PULSE WIDTH, t_p	μs	0.9	1.0	1.1
PULSE INTERVAL, t_i	ms	2.0	3.0	4.0
PULSE VOLTAGE, V_p	V	NONE	0.22	0.4
QUIESCENT VOLTAGE, V_Q	V	2.4	5.0	NONE



Example 3:

Stimulus - level 5; response - level 3

Fault locator, TP 24

- Part 1

  A. Stimulus

  1. Apply 26.3 VDC for 10 msec to the connections 1A3P1-16 and 1A3P1-15.
  2. Reduce the 5 VDC standby voltage to 4.45 VDC within 100 usec at the connections 1A3P1-13 and 1A3P1-35.

  B. Response

  1. Test signal goes off anytime the voltage at 1A3P1-13 and 1A3P1-35 is between 4.45 and 4.55.

- Part 2

  A. Stimulus

    1. Apply 5 VDC to connections 1A3P1-13 and 1A3P1-35.
    2. Apply 26.3 VDC for 10 msec to the connection 1A3P1-16 and 1A3P1-15.
    3. Increase the 5 VDC standby voltage to 5.55 VDC within 100 usec at the connections 1A3P1-13 and 1A3P1-35.

  B. Response

    1. Test signal goes off anytime the voltage at 1A3P1-13 and 1A3P1-35 is between 5.45 and 5.55 VDC.

Example 4:

Stimulus - level 1; response - level 1
Fault locator, TP 18

- Part 1

  A. Stimulus

    1. Ground connections 1A3P1-67 and 1A3P1-35.

  B. Response

    1. Wait 200 msec, then perform test.
    2. 5 ohms max., 1A3P1-69 and 1A3P1-35

- Part 2

  A. Stimulus

     1. Open connections 1A3P1-67 and 1A3P1-35.

  B. Response

     1. Wait 200 msec, then perform test.
     2. 100K ohms min., 1A3P1-69 and 1A3P1-35

Example 5:

Stimulus - level 2; response - level 1
Fault locator, TP 38

- Part 1

  A. Stimulus

     1. Apply a 1 usec negative going pulse (logic 0) to the connection 1A3P2-5 and 1A3P2-26.

  B. Response

     1. Test signal is on, logic 0 for connections 1A3P2-6 and 1A3P2-47.

- Part 2

  A. Stimulus

     1. Apply a 1 usec negative going pulse (logic 0) to the connections 1A3P2-4 and 1A3P2-25.

  B. Response

     1. The test signal is off, logic 1 for connections 1A3P2-6 and 1A3P2-47.

Example 6:

Stimulus - simple; response - level 6
Fault locator, TP 41

A. Stimulus

1. Apply 26.3 VDC to connections 1A3P1-16 and 1A3P1-15.

B. Response

1. Test is performed at UUT timing rate.
2. B20 of all 512 words are tested.
   a. Enable - logic 1 is 0.4 VDC max.
   b. Inhibit - logic 0 is 2.4 VDC min.
3. Data must be sampled and read within 6 to 8 usec after ROM clock pulse.

Example 7:

Stimulus - level 2; response - level 3
RF amp TP 85

- Part 1

A. Stimulus

1. Apply 0.3 VDC at connections 2J2-59 and 2J2-60.
2. Apply a 100 Hz square wave with 50% duty cycle and amplitude 0.3 VDC at connections 2J2-52 and 2J2-51.
3. Apply CW RF at frequencies (F5 + F2)/2 (band 1) and (F9 + F4)/2 (band 2).

a. Apply -14.7 dBm at band 1.

b. Apply -17.4 dBm at band 2.

B. Response

1. Check depth of modulation at connections 2J5, 2J6-P, and 2J6-S.

a. For band 1, the depth of modulation is 13 dB ± 1.8 dB below P1C.

b. For band 2, the depth of modulation is 13 dB ± 1.8 dB below P2C.

- Part 2

A. Stimulus

1. Repeat part 1 stimulus using frequencies F2 (band 1) and F4 (band 2).

B. Response

1. Repeat part 1 response using frequencies F2 (band 1) and F4 (band 2).

- Part 3

A. Stimulus

1. Repeat part 1 stimulus using frequencies F5 (band 1) and F9 (band 2).

B. Repeat part 1 response using frequencies F5 (band 1) and F9 (band 2).

Example 8:

Stimulus - level 3; response - level 4

Fault locator, TD 40-2

A. Stimulus

1. Apply 26.3 VDC to the connections 1A3P1-16 and 1A3P1-15.
2. Apply logic 1 to connections 1A3A2P1-13 and 1A3A2P1-1.
3. Apply 47 negative going pulses (logic 0) at the connections 1A3A2P1-9 and 1A3A2P1-1.
   a. The pulse train should have a period of 3 usecs with a pulse width of 1 usec.

B. Response

1. Monitor the counter at connections 1A3P1-96 through 1A3P1-104 and 1A3P1-35.
   a. The counter will count to 47 and stop.

APPENDIX C

ATLAS TEST PROGRAM FLOW DIAGRAM EXAMPLES

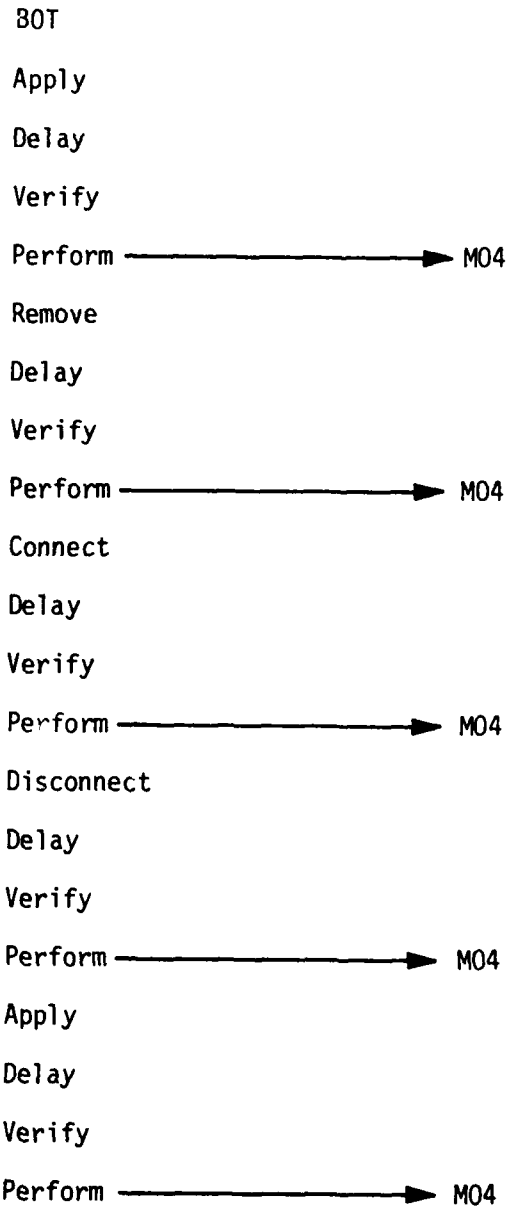


Figure C-1. FL TP 13 (S = 2, R = 2)

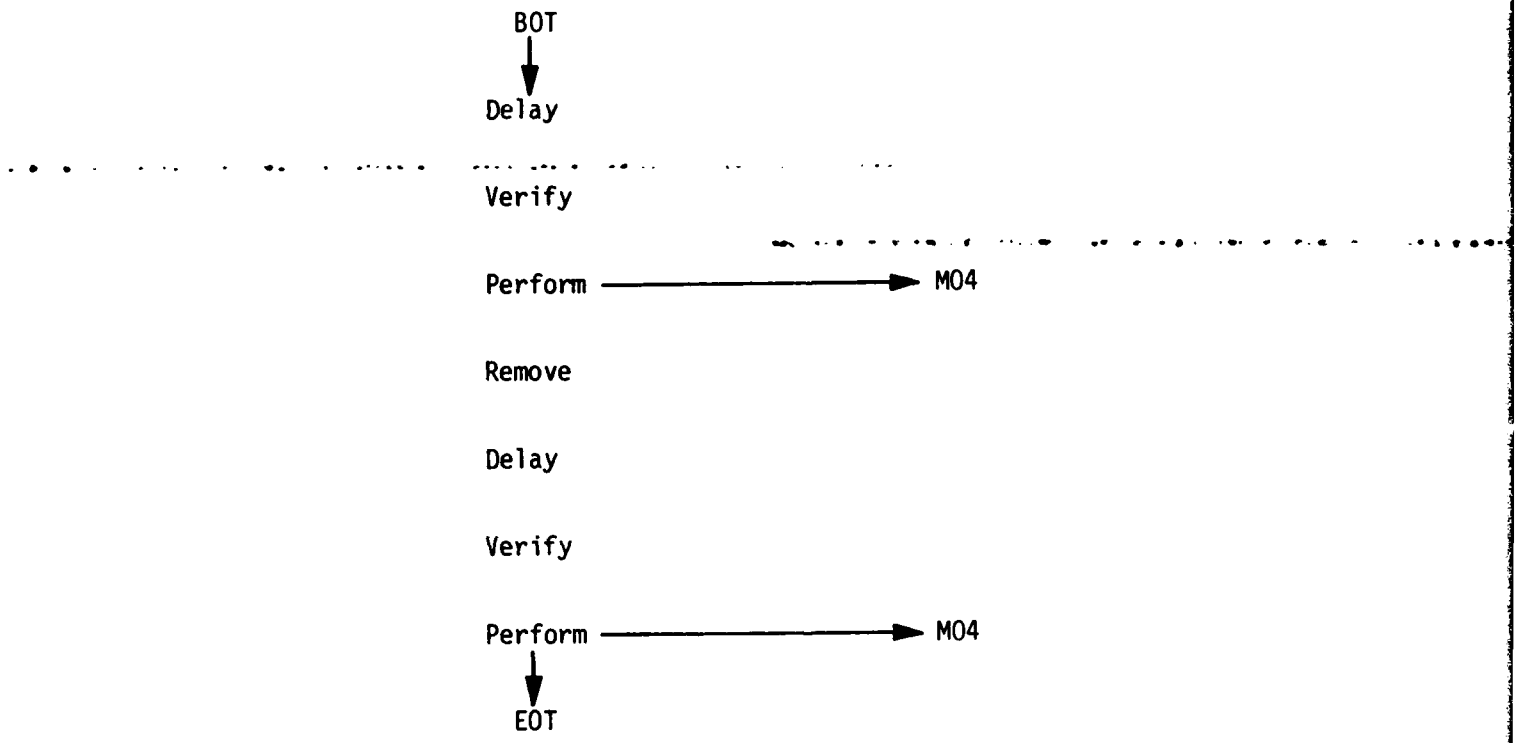


Figure C-2. FL TP 17 (S = 1, R - 1)

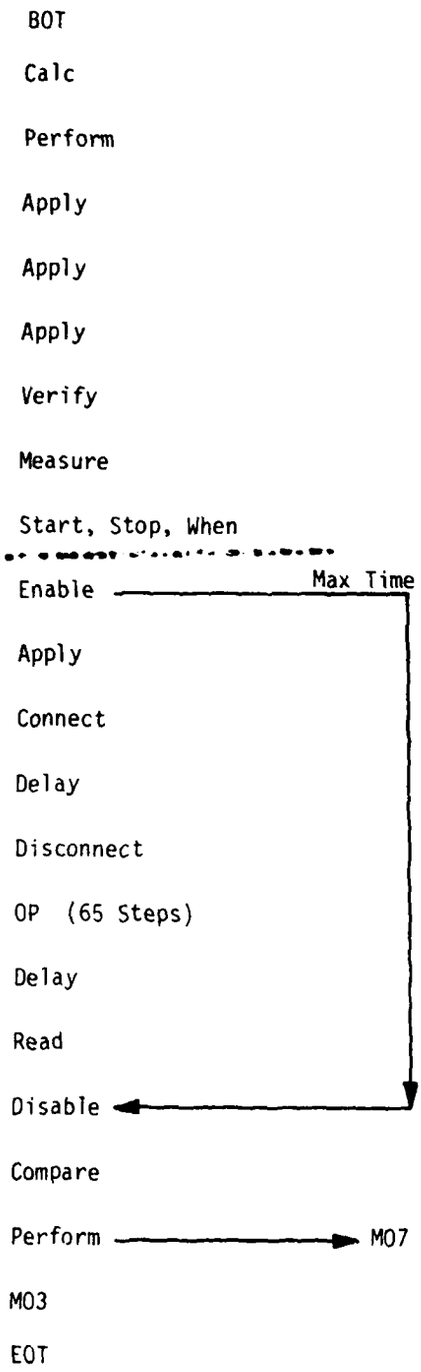


Figure C-3. FL TP 23.2 (S = 3, R = 3)

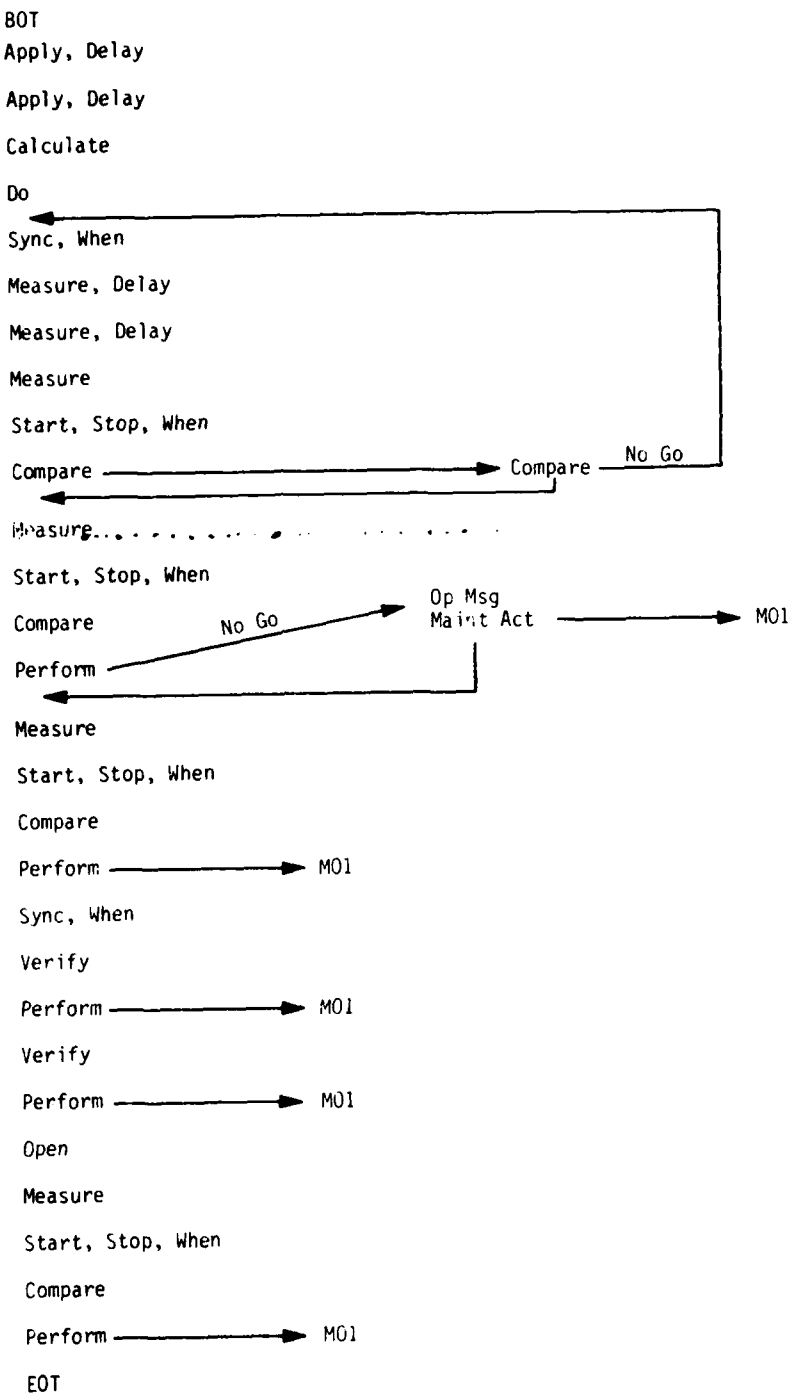


Figure C-4. FI. TP 32 (S = 1, R = 8)

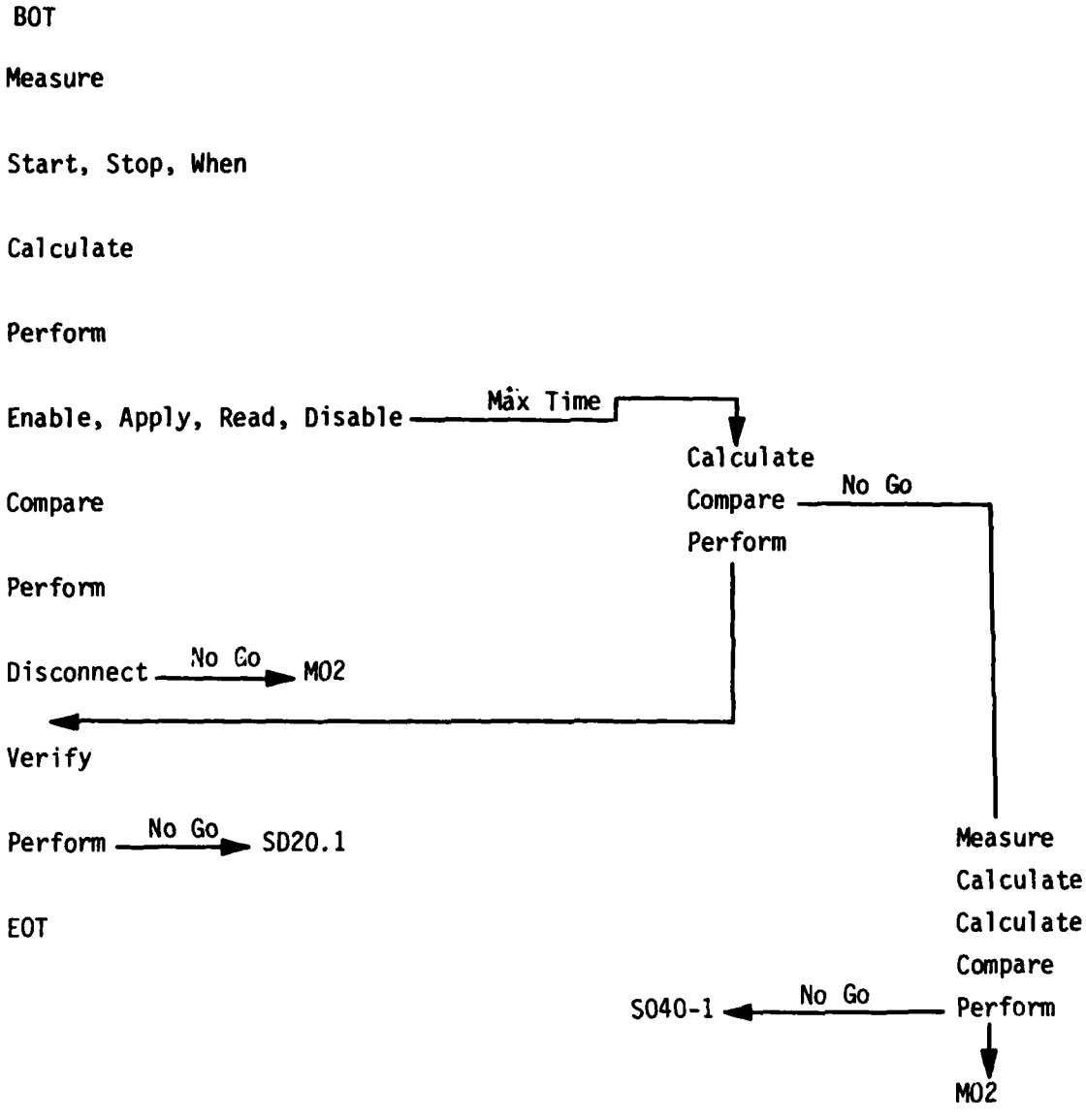


Figure C-5. FL TP 39 (S = 0, R = 6)

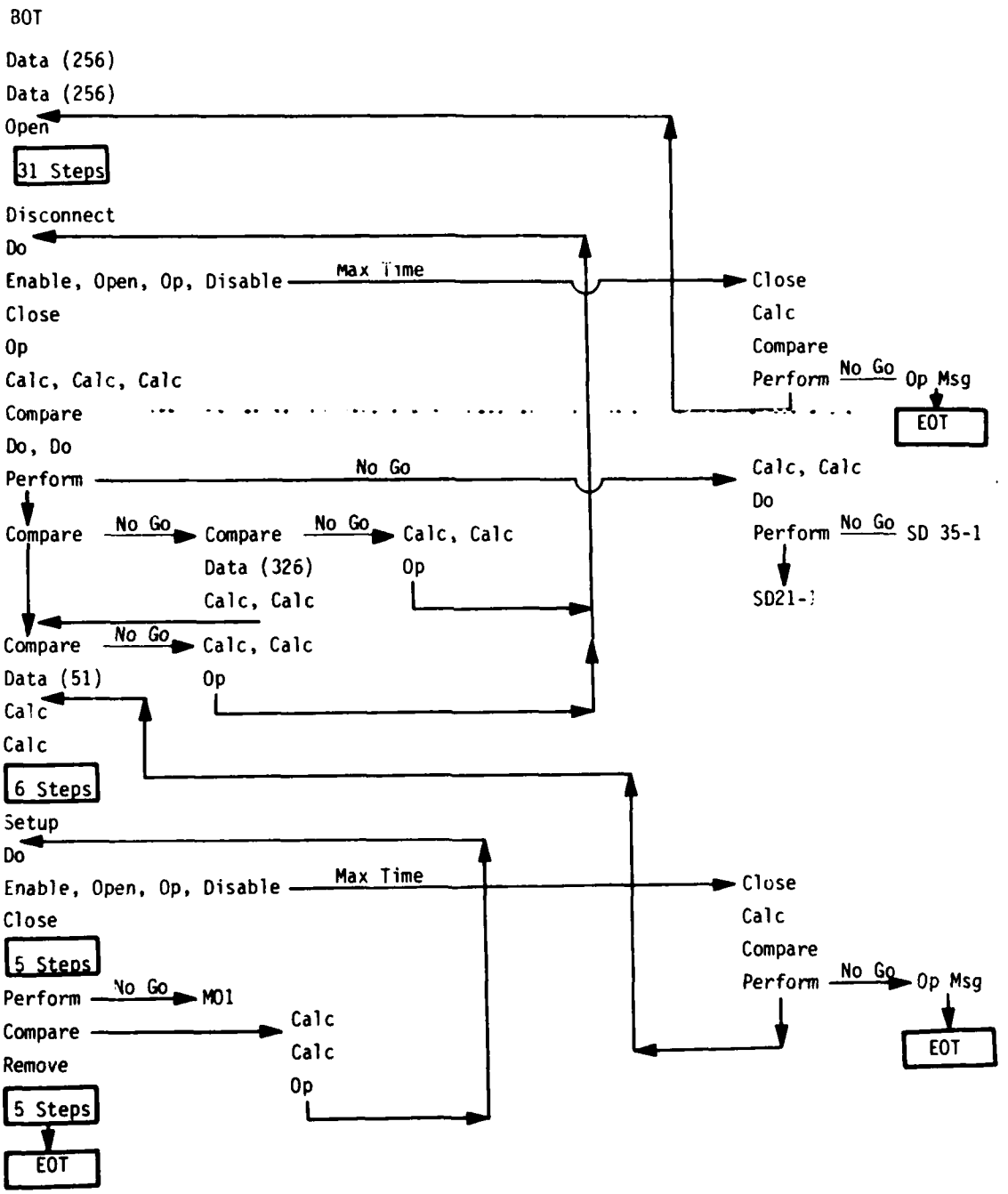
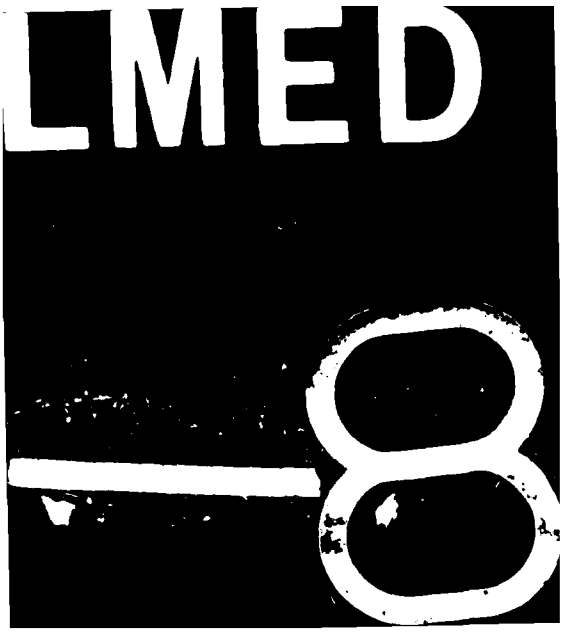


Figure C-6. FL TP 47 (S = 2, R = 7)

☆U.S.Government Printing Office: 1980 — 657-084/109


LMED
8